Universidad Zaragoza
1542

# Trabajo Fin de Máster

## Detección de actividades humanas con sensores portables

## Detection of human activities with wearable sensors


Autor

Luis Benages Pardo

Director

Julio David Buldain Pérez

ESCUELA DE INGENIERÍA Y ARQUITECTURA


2018

TRABAJOS DE FIN DE GRADO / FIN DE MÁSTER

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

# DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe acompañar al Trabajo Fin de Grado (TFG)/Trabajo Fin de Máster (TFM) cuando sea depositado para su evaluación).

D./Dª. Luis Benages Pardo,

con nº de DNI 73105028B en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza,

Declaro que el presente Trabajo de Fin de (Grado/Máster) Máster, (Título del Trabajo) Detección de actividades humanas con sensores portables,

es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada debidamente.

Zaragoza, 19 de Noviembre de 2018

Fdo: Luis Benages Pardo

# AGRADECIMIENTOS

Agradecer a David Buldain toda la ayuda y tiempo prestados durante el desarrollo de este proyecto, lo cual ha hecho posible la comprensión y adquisición de una gran cantidad de nuevos conocimientos.

También agradecer a Marta, Paula, Lucía, Paola, Jose, Guille y Baigorri por el tiempo y esfuerzo dedicados durante la recopilación de la base de datos con la que se ha desarrollado este proyecto.

# RESUMEN


DETECCIÓN DE ACTIVIDADES HUMANAS CON SENSORES PORTABLES

La recopilación de datos y estadísticas deportivas es un campo de investigación en continuo desarrollo. Actualmente, existe una gran demanda en el mercado de dispositivos portables capaces de recopilar información importante durante el desempeño de cualquier actividad. Esa información es procesada para obtener estadísticas o retroalimentación con la que mejorar la técnica deportiva de los usuarios.

En este trabajo de fin de máster se va a diseñar e implementar un sistema capaz de distinguir entre distintas actividades realizadas durante el desempeño de un partido de tenis, siendo la finalidad última la correcta clasificación de un conjunto de golpes realizados. Además, el modelo ha de presentar robustez ante la variabilidad de las dimensiones, edad o sexo de cualquier sujeto que realice las actividades. Se desarrollará una base de datos propia con el fin de estar orientada al objetivo propuesto, y así poder incluir nuevos movimientos en ampliaciones futuras al trabajo. Para ello, se ha seleccionado y configurado dos nodos sensores que utilizan la tecnología inalámbrica de bajo consumo *Bluetooth Low Energy* para comunicarse con un PC que actúa como dispositivo central para recopilar la información recibida por los sensores.

Para conseguir estos objetivos, se han procesado los datos de los sensores mediante el cálculo de sus espectrogramas, de manera que aplicando técnicas novedosas de *Deep Learning* con entrenamiento semi-supervisado se consiga llevar a cabo la extracción de características y clasificación de los espectrogramas obtenidos. El aprendizaje semi-supervisado permite que la cantidad de datos etiquetados necesaria se reduzca considerablemente a cambio de obtener una gran cantidad de datos no etiquetados. La implementación de estas técnicas se ha llevado a cabo mediante el uso de TensorFlow con su API en Python.

Una vez entrenados los sistemas clasificadores, se ha comprobado su funcionamiento sobre nuevos datos de test para así comprobar la correcta clasificación de las actividades sobre datos no vistos por el clasificador durante su entrenamiento. De este modo, se demuestra la capacidad generalizadora del modelo.

# Índice de contenido

# Índice de figuras

# Índice de tablas

# Capítulo 1
# Introducción

## 1.1. Motivación

Este trabajo se enmarca en el proyecto de investigación: "Análisis semántico y comprensión del comportamiento del jugador en aplicaciones deportivas", financiado por el MINECO (TIN2017-88841-R) y cofinanciado por fondos FEDER. Concedido en 2018 al grupo de investigación CVLab (departamento de ingeniería electrónica y comunicaciones).

La recopilación de datos y estadísticas durante el desempeño de cualquier deporte es un tema de investigación en continuo desarrollo, ya que tanto personas aficionadas como profesionales buscan una manera de obtener información sobre su rendimiento o posibilidad de mejora. Debido a ello, se desarrolla este proyecto en el ámbito de reconocimiento de actividades realizadas en un partido de tenis, al que se añadirá en un futuro nuevas funcionalidades con las que poder mejorar el rendimiento deportivo de las personas.

## 1.2. Objetivos

El objetivo de este proyecto es el llevar a cabo la clasificación de un conjunto de actividades humanas intentando solventar los problemas de generalización del modelo que se suelen encontrar en los sistemas habituales de reconocimiento de actividades humanas con sensores portables. Como ya se ha comentado en la motivación del proyecto, la finalidad principal es el desarrollo de un modelo clasificador que consiga distinguir un conjunto de golpes y movimientos que se pueden realizar durante un partido de tenis.

Para ello, la solución propuesta aplica técnicas de Deep Learning con modelos convolucionales (CNNs) de aprendizaje semi-supervisado para extraer información de los espectrogramas de las señales recopiladas por dos nodos sensores portables. Estos sensores están colocados en la muñeca y cintura del sujeto, transmitiendo información de acelerómetro y giróscopo en sus tres ejes de coordenadas.

Con la extracción de características de los espectrogramas utilizando CNNs se pretende que el modelo sea robusto ante las diferencias de tamaño, edad, o sexo de los distintos sujetos que realicen las actividades. La aplicación de CNNs para la extracción de características introduce invariancia ante la posición y deformaciones de éstas en el espectrograma.

De este modo, aunque un mismo movimiento realizado por diferentes usuarios presente frecuencias distintas en el espectrograma, las formas obtenidas serán las mismas, pero desplazadas en frecuencias, por lo que las CNN serán capaces de generar representaciones de características en su salida invariantes ante estas fuentes de variabilidad.

Para la construcción del modelo se utilizarán las librerías de TensorFlow con la API desarrollada en Python, ya que es la versión más estable que presenta y con más documentación. De esta manera se abre la posibilidad de utilizar con software libre una gran cantidad de herramientas desarrolladas en Python.

Para el entrenamiento y test del sistema de reconocimiento de actividades planteado se va a crear una base de datos propia. En ella habrá una gran cantidad de datos sin etiquetar para el entrenamiento del modelo convolucional que realizará la extracción de características, y unos pocos datos etiquetados para poder entrenar el clasificador supervisado que llevara a cabo el reconocimiento del conjunto de actividades. La base de datos la realizarán 8 personas distintas, 4 de cada sexo, para así poder obtener un modelo generalizado. Se habrá de seleccionar y configurar dos nodos sensores que cumplan con los principales requisitos de portabilidad y bajo consumo.

De esta manera, los pasos a seguir para el cumplimiento de los objetivos planteados son los siguientes:

- Estudio del estado del arte y aprendizaje sobre el uso las distintas herramientas utilizadas.
- Búsqueda de posibles sensores, aprendizaje de su tecnología, y configuración de estos.
- Generación de una base de datos con la que entrenar y testear el modelo.
- Procesamiento de espectrogramas de las señales
- Entrenamiento de redes convolucionales de manera no supervisada para extracción de características, y entrenamiento de clasificador de manera supervisada. Ajuste de hiperparámetros.
- Análisis de resultados y conclusiones.

## 1.3. **Herramientas utilizadas**

Para el desarrollo del proyecto, se ha hecho uso de las siguientes herramientas:

- Herramientas y dispositivos de Texas Instruments: Sensores **SensorTag CC2650**, junto con **debugger DevPack**, **CC2540 USB dongle**, y software **BTool**, para la captación de datos y comunicación con el PC.
- **Code Composer Studio 8.1.0:** para programación de los sensores con el debugger DevPack.
- **MATLAB R2017b:** para el tratamiento de ficheros de texto en la lectura de datos enviados por el sensor.

- **Spyder 3.2.8:** para programación en **Python,** donde se han construido las redes con librerías de **TensorFlow** y **scikit-learn**. Tarjeta gráfica de PC utilizado: **NVIDIA GEFORCE GTX1050.** Memoria RAM: 8GB.
- **Google Colaboratory**: plataforma que te permite de manera gratuita un entorno de ejecución virtual, con **12GB de RAM** y una tarjeta gráfica **Tesla K80 GPU**. Esta plataforma se ha utilizado para el entrenamiento de todas las redes.

## 1.4. Contexto y estado del arte

El reconocimiento de actividades humanas (RAH) es un campo de investigación en continuo crecimiento desde la aparición de sensores capaces de captar información del entorno durante el desempeño de cualquier actividad. Desde la década de los noventa, se ha estado ampliando el estado del arte en este campo mediante el desarrollo de nuevas técnicas que permiten obtener una mayor precisión en la clasificación de actividades a partir de sensores.

Muchas de estas técnicas utilizan el denominado aprendizaje automático o *machine learning* (ML) basado en modelos de clasificación, los cuales se obtienen mediante procesos de aprendizaje a partir de ejemplos. El entrenamiento de los clasificadores se lleva a cabo con una gran cantidad de ejemplos, ya sea mediante aprendizaje supervisado, o no supervisado. En el aprendizaje supervisado se entrena el clasificador mediante datos etiquetados, de manera que se le indica al clasificador la salida deseada ante unos datos de entrada determinados. En el caso del aprendizaje no supervisado, el clasificador va agrupando los datos que son similares entre sí en base a una medida de distancia (*clustering*). Una vez se dispone del clasificador entrenado, se realiza el test de dicho clasificador con nuevos datos con los que no haya tratado todavía, de manera que según los resultados que se obtienen con esos nuevos datos de test se comprueba la eficacia del modelo de clasificación obtenido.

Hoy en día, el desarrollo de nuevos sensores de menor tamaño y menor consumo está permitiendo que la recopilación de datos mediante sensores portables o externos se realice de manera mucho más sencilla, y sin interferir en las actividades realizadas. La información obtenida por estos sensores es de gran interés en una amplia variedad de aplicaciones, como por ejemplo en aplicaciones médicas para el seguimiento de pacientes con enfermedades que necesitan monitorización [1], en la rehabilitación de enfermedades o lesiones mediante el seguimiento del progreso del paciente [2] [3] [4], en la recopilación de estadísticas deportivas para la visualización del progreso realizado [5], o en la mejora de la técnica en el desempeño de ciertos deportes [6] [7].

Aunque el objetivo último de diferentes investigaciones sea el mismo (clasificar las mismas actividades que llevan a cabo diversas personas), los métodos que se utilizan en cada investigación para llevarlo a cabo pueden ser distintos. Cada sistema de RAH puede utilizar distintos tipos de sensores, diversos métodos de extracción de características, variadas tipologías de clasificadores, o diferentes métodos de entrenamiento del clasificador. Sin

embargo, el diagrama de flujo que siguen habitualmente la estructura de todos ellos es el mismo [8] [9], tal y como se muestra en la Fig. 1.1.

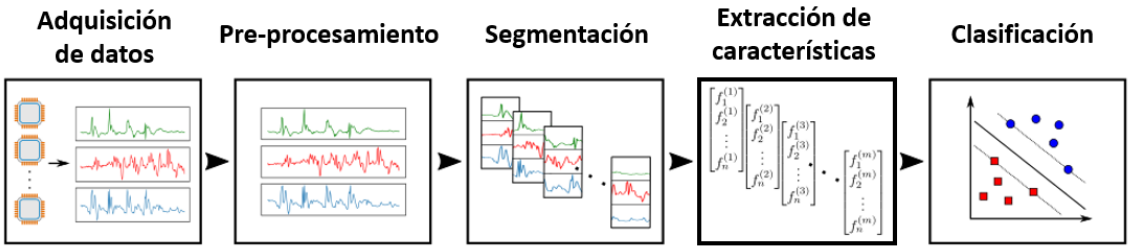



*Fig. 1.1. Diagrama de bloques típico de un sistema de RAH*

### 1.4.1. Adquisición de datos

El primer paso a la hora de realizar un sistema clasificador de actividades es la obtención de una base de datos con la que entrenar dicho clasificador. La opción más común en sistemas de RAH para una aplicación concreta es la elaboración de una base de datos propia, aunque también se pueden utilizar bases de datos ya realizadas en las que se encuentren una gran cantidad de datos. Para ello, se ha de seleccionar tanto el tipo de sensor utilizado como la localización del sensor en el cuerpo, ya que ambas opciones de diseño influyen en los resultados de la clasificación.

Durante el estudio del estado del arte, se han encontrado gran variedad de sensores utilizados en distintas investigaciones. Una opción es la detección de actividades mediante el tratamiento de imágenes de video [10] con cámaras colocadas en el entorno donde se encuentra el sujeto. Sin embargo, esta opción sólo es apta para cierto tipo de aplicaciones ya que suelen ser caras y conllevan problemas de privacidad y de elevado coste computacional en el tratamiento de las imágenes. La opción más utilizada a la hora de seleccionar sensores para RAH es el uso de sensores portables, debido a que las tecnologías actuales de estos dispositivos presentan prestaciones de bajo consumo y pequeñas dimensiones [11]. Estas prestaciones los hacen idóneos para la recopilación de información durante largos periodos de tiempo, sin interferir en las actividades realizadas, y en cualquier entorno.

Existen distintos tipos de sensores portables utilizados para sistemas de RAH, como acelerómetros, giróscopos, magnetómetros, sensores de temperatura, barómetros, localizadores, micrófonos, pulsómetros o electrocardiógrafos. Sin embargo, la mayoría de aplicaciones investigadas en el estado del arte utilizan únicamente acelerómetros y giróscopos ya que la información que proporcionan es suficientemente relevante para la clasificación de un gran número de actividades [12] [13] [14]. La localización en la que se ubican estos sensores en el cuerpo durante el desempeño de las actividades a clasificar es también un parámetro de diseño en investigación [15]. El cambio de la ubicación del sensor en el cuerpo hace que la información obtenida por los sensores varíe, de manera que los datos obtenidos pueden ser más o menos representativos de las actividades realizadas.

En el uso de sensores portables, la recopilación de los datos obtenidos por los distintos sensores se realiza habitualmente mediante tecnologías

inalámbricas, utilizando como dispositivo maestro un móvil smartphone o un ordenador portátil conectado a distintos sensores esclavos a través de un protocolo de bajo consumo, como por ejemplo *Bluethooth Low Energy* (BLE), o *ZigBee* [4] [11].

En el caso de querer utilizar una base de datos ajena, en [16] [17] [18] se muestran las distintas bases de datos más utilizadas para entrenar o validar clasificadores de RAH.

La evaluación de los clasificadores que se desarrollen se ha de llevar a cabo mediante datos que el clasificador no haya utilizado durante el entrenamiento, de manera que se debe realizar una partición de la base de datos de la que se dispone antes de entrenar el clasificador.

En casos en los que se tenga un gran conjunto de datos, se realiza la separación de estos en tres subconjuntos: entrenamiento, validación y test. El modelo se entrena únicamente con los datos de entrenamiento. El conjunto de validación se utiliza para mejorar los resultados obtenidos con modelos entrenados mediante el ajuste de los hiperparámetros de diseño del modelo. El conjunto de test se utiliza únicamente una vez un modelo ha sido entrenado con el conjunto de hiperparámetros que generaron los mejores resultados posibles con el conjunto de validación. Este tipo de separación de la base de datos se utiliza principalmente para mejorar los resultados obtenidos evitando el sobreajuste del modelo a los datos de entrenamiento, así como para explorar el conjunto de hiperparámetros óptimos para el entrenamiento final del modelo.

En el caso de disponer de pocos datos, debido a la posibilidad de sesgos en el reparto de los ejemplos en los distintos subconjuntos de datos, se suele utilizar validación cruzada, que consiste en dividir la base de datos en varios subconjuntos y entrenar tantos modelos como subconjuntos haya, utilizando en cada uno de esos modelos un subconjunto distinto de validación. Finalmente se promedia los resultados obtenidos con cada una de las redes de la validación cruzada. De esta manera los parámetros del modelo no se pueden ajustar a un conjunto de datos de evaluación concreto, evitando que los resultados presenten sobreajuste a un conjunto de datos particular.

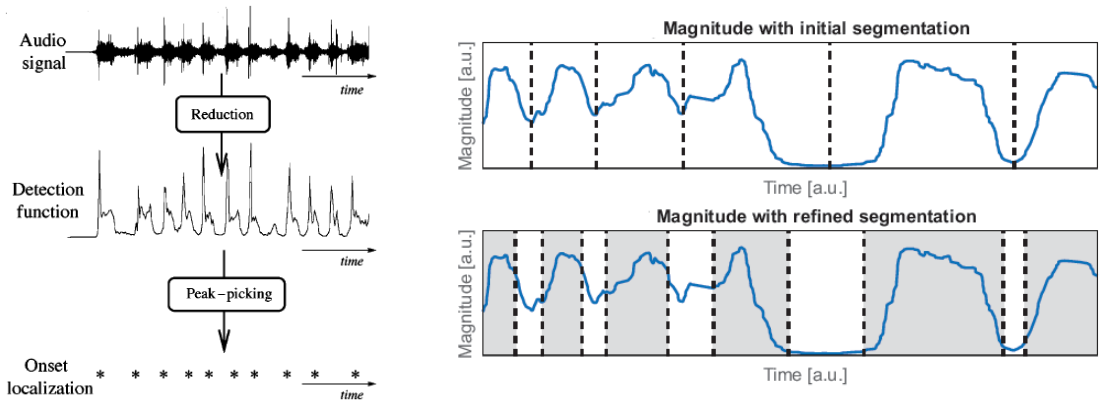
### 1.4.2. **Preprocesamiento y segmentación**

Una vez se dispone de una base de datos con la que poder entrenar un clasificador, es habitual encontrar en el estado del arte ciertos tipos de preprocesamiento de los datos.

Los datos obtenidos del sensor suelen ser filtrados para reducir el ruido introducido en la medición del sensor [6]. También es común realizar un escalado de los datos, para que los rangos de medida de distintos tipos de sensores no influyan en la mayor ponderación de uno u otro durante el entrenamiento. En ocasiones, se lleva a cabo sincronización de las muestras de distintos sensores que presentan diferentes frecuencias de muestreo. Para la reconstrucción de datos perdidos durante el muestreo, se realiza una interpolación del valor en el instante anterior y posterior al dato perdido [6] [19].

La recopilación de datos de los sensores para RAH se realiza durante largos periodos de tiempo, obteniendo una gran cantidad de muestras de cada sensor. La extracción de características directamente sobre largos periodos de señales temporales, dificultaría enormemente la obtención de características discriminativas de las distintas actividades realizadas durante ese periodo. Por ello, en la mayoría de los casos se lleva a cabo una segmentación de la señal que permita obtener pequeños periodos de la señal donde pueda ocurrir una cierta actividad en lugar de varias. De este modo, se consigue extraer características más discriminativas de cada actividad para poder llevar a cabo su clasificación. En el caso de señales temporales, los dos métodos de segmentación principales son la segmentación por umbral, y la segmentación mediante ventanas temporales deslizantes [9]. La segmentación por umbral se utiliza principalmente en aplicaciones en las que se está buscando instantes temporales concretos de la señal, como instantes en los que hay máximos o mínimos locales, o instantes donde la energía de la señal supera un valor concreto.

Un ejemplo es la segmentación realizada en [20], donde se lleva a cabo la segmentación de señales musicales para detectar el inicio de transición entre notas. Otro ejemplo es el encontrado en [7] donde se desea realizar clasificación de golpes de tenis, por lo que se lleva a cabo la extracción de ventanas temporales de únicamente los instantes donde se produce un golpe. En la Fig. 1.2 se muestran los dos ejemplos citados.




*a)* *b)*

*Fig. 1.2. Segmentación por umbral. a) Segmentación de una señal de audio. b) Segmentación de señal para clasificación de golpes de tenis.*

La segmentación por umbral no es el método más común, ya que en la mayoría de las aplicaciones produce la pérdida de información relevante para una correcta clasificación. La técnica de segmentación más utilizada en RAH es la segmentación mediante una ventana temporal deslizante. Este método extrae pequeños segmentos de la señal mediante una ventana temporal de anchura definida, deslizándola a lo largo de toda la señal. De este modo, se consigue tratar con pequeños intervalos de la señal, sin perder información alguna sobre los datos de entrada. Habitualmente el desplazamiento de la ventana a lo largo de la señal se realiza con una cierta superposición respecto de la ventana anterior (50% o más normalmente) para así evitar perder información en las transiciones de una ventana a otra y aumentar la redundancia de las muestras temporales extraídas. En la Fig. 1.3 se observa un ejemplo de segmentación con ventana deslizante para una aplicación de

electromiografía desarrollada en [21]. Es común utilizar la técnica de enventanado de Hanning para evitar así las distorsiones producidas por las discontinuidades en los límites de la ventana.

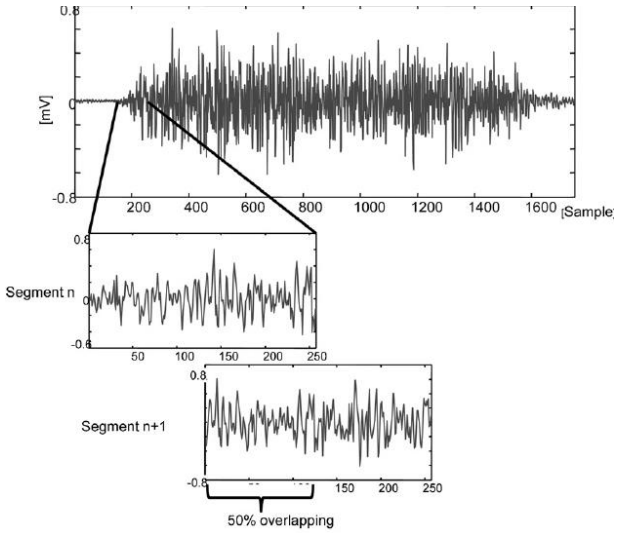



*Fig. 1.3. Segmentación mediante ventana temporal deslizante*

La anchura de la ventana deslizante es un parámetro de diseño típico en sistemas de RAH ya que influye en los resultados de la clasificación final realizada. En [22] se realiza un estudio sobre el impacto de la anchura de la ventana deslizante en los resultados obtenidos en la clasificación, donde se concluyó que una ventana de uno a dos segundos es el tamaño óptimo para conseguir una buena relación entre la precisión obtenida y la velocidad de segmentación de la señal completa. También se han realizado estudios acerca de la posibilidad de desarrollar un método de segmentación mediante una ventana deslizante de anchura adaptativa [23].

### 1.4.3. **Extracción de características**

La extracción y selección de características representativas de la señal es uno de los procesos más importantes a la hora de desarrollar un sistema de RAH. En el campo de ML a este proceso se le denomina la fase de *feature engineering*, consistente en una fase de *feature extraction* seguida de una fase de *feature selection*. Tras muchos años de investigación en este campo, se ha comprobado que el uso de estas características como datos de entrada a un clasificador mejora notablemente los resultados obtenidos en la clasificación respecto al uso directo de los valores muestreados por el sensor. En [24] se lleva a cabo la comparación de los resultados obtenidos en una aplicación que utiliza inicialmente los datos del sensor muestreados, y los obtenidos cuando se utilizan las características extraídas como datos de entrada. Además de mejorar los resultados obtenidos, la extracción de características permite llevar a cabo una reducción en la dimensionalidad del problema.

### Extracción de características prediseñadas

Este tipo de extracción consiste en el cálculo de un conjunto amplio de características en el dominio temporal y frecuencial que se consideren suficientemente representativas de la señal. Con las características seleccionadas se forman distintos grupos de ellas para comprobar así con cuál de ellos se obtiene un mejor resultado tras la clasificación. En la Tabla I se muestran las características comúnmente utilizadas en el estado del arte de RAH [1], [6], [8], [15], [25].

*TABLA I. Características seleccionadas habitualmente en RAH*

| **Dominio temporal** | | **Dominio frecuencial (FT)** |
|---|---|---|
| Media | Integral | Componente DC |
| Mediana | Derivada | Suma de coeficientes |
| Varianza | Cruces por cero | Frecuencia dominante |
| Desviación estándar | Pico a pico | Energía espectral |
| Mínimos y máximos | Kurtosis | Pico de frecuencia |
| Amplitud | Distancia euclídea | Entropía espectral |
| RMS | SMA | Potencia |
| Correlación | SVM | |
| Correlación cruzada | DSVM | |

Existen también distintas técnicas para comprimir un gran conjunto de características a un subconjunto transformado con características más representativas para la clasificación de los conjuntos de datos. Este tipo de técnicas consiguen reducir la dimensionalidad del problema, siendo las más comunes el análisis de componentes principales (PCA), análisis lineal discriminante (LDA), o el análisis de componentes independientes (ICA) [26].

Aunque el cálculo de características prediseñadas es el método más encontrado en el estado del arte, los modelos de RAH desarrollados de esta forma presentan grandes problemas de generalización [16] [18]. Esto se debe a que generalmente los investigadores consiguen diseñar un conjunto de características que les permita obtener unos buenos resultados, pero validando el modelo con la misma base de datos para la que se diseñaron dichas características. Sin embargo, esta situación se daría únicamente en un escenario de laboratorio ya que, en una aplicación real, las actividades serán realizadas por usuarios de distintas dimensiones respecto a las personas que recopilaron los datos utilizados para el entrenamiento. La diferencia corporal entre los sujetos introduce mucha variabilidad en la información recogida por los sensores, llegando a provocar que la varianza de la señal debido a las diferencias corporales sea mayor que la varianza detectable entre las diferentes actividades. Esta situación hace que los clasificadores presenten sobreaprendizaje sobre la muestra de ejemplos corporales de la base de datos, no siendo capaces de generalizar a nuevas señales procedentes de sujetos con dimensiones corporales distintas a las presentes en la base de datos. Por ello, la precisión obtenida en la clasificación de actividades en una aplicación real es mucho menor que la

obtenida con los usuarios con los que se realizó la base de datos [9] [16] en el caso de utilizar características prediseñadas.

**Extracción automática de características**

Actualmente, los investigadores están intentando llevar a cabo la extracción automática de características relevantes de los datos de entrada. De este modo, aunque el conjunto de datos de entrada cambie, el modelo seguiría obteniendo características suficientemente representativas de los datos para obtener unos buenos resultados en la clasificación. La técnica más habitual en RAH para extracción automática de características se basa en la aplicación de Deep Learning (DNN) a los datos de entrada, siendo las redes neuronales convoluciones (CNN) la más utilizadas [18].

Las CNN son populares para procesamiento de imágenes ya que aprenden automáticamente los filtros necesarios para extracción de patrones [27] [28]. De este modo, las CNN aprenden características discriminativas de los datos de entrada, presentando invariancia ante la posición, escala y orientación de éstas. Esta cualidad las hace idóneas para muchas aplicaciones de reconocimiento de patrones en imágenes.

Este tipo de redes también son utilizadas para extracción de características en señales temporales como las proporcionadas por sensores portables [29] [30] [31] [32] [33]. Para ello, se suele llevar a cabo una representación 2D de dichas señales como si de una imagen se tratara, de manera que las coordenadas $x$ e $y$ de dicha representación se asocien a índices de pixeles en una imagen [34].

Existen distintas técnicas con las que transformar los datos de entrada para generar una representación 2D de los mismos e introducirlos así a la CNN. Como ejemplo, se puede utilizar directamente la señal temporal como una representación 2D [31], apilar todas las señales de los distintos sensores y aplicarles la FFT [32] o realizar el cálculo del espectrograma de las señales [30] que permite extraer características de la evolución frecuencial a lo largo del tiempo. Otra forma menos utilizada para aplicar CNN a señales temporales es obtener cada dimensión de un sensor como un canal independiente (del mismo modo que se realiza en una imagen RGB) y aplicar convolución en 1D a cada canal [18].

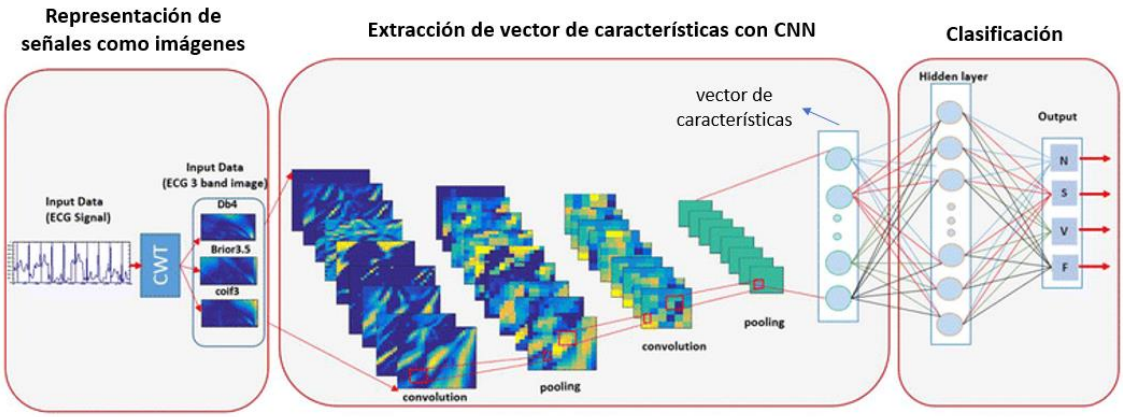



*Fig. 1.4. Extracción de características mediante CNN*

En la Fig. 1.4. se muestra un ejemplo de una CNN utilizada para extraer características de un señal de electrocardiograma [35]. Tal y como se ha

comentado en el párrafo anterior, se comienza transformando la señal de entrada a una representación 2D como una imagen. Después se aplica la CNN a esa imagen, consiguiendo a su salida un vector de características extraídas de los datos de entrada. Por último, las características son introducidas a un clasificador supervisado que realiza el reconocimiento de los patrones extraídos y los clasifica en una de las clases etiquetadas.

Otro tipo de DNN utilizadas para extracción de características representativas de los datos de entrada son los *Autoencoders*. Este tipo de DNN va aprendiendo en sus capas ocultas una codificación de los datos de entrada mediante la replicación de la entrada en la salida de la red, utilizando datos sin etiquetar. Una vez entrenada la red, la codificación obtenida en la capa intermedia tiene información suficiente como para generar una salida de una clase determinada con la que se ha entrenado. De este modo, la capa intermedia del *Autoencoder* contiene una codificación representativa, y generalmente comprimida, del dato introducido a la red. Esta codificación es una representación similar al vector de características obtenido en las CNN, a partir del cual se puede entrenar al clasificador de salida. En la Fig. 1.5 se puede observar la estructura típica que tienen este tipo de redes.

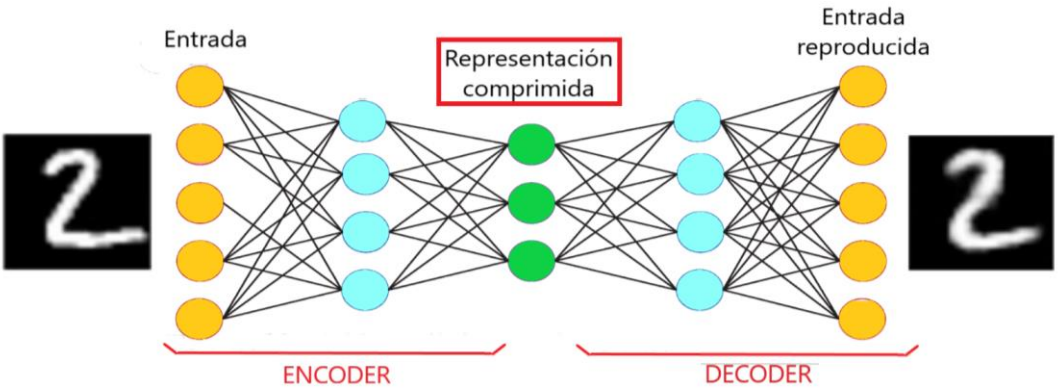



*Fig. 1.5. Estructura de un Autoencoder*

### 1.4.4. Clasificación

Durante la revisión del estado del arte, se han encontrado numerosos métodos de clasificación utilizados para RAH. En la Tabla II se muestran los clasificadores convencionalmente utilizados en aplicaciones de RAH [8] [9] [24], indicando los más usados dentro de cada tipología de clasificadores. Como se puede observar se utilizan tanto clasificadores con entrenamiento supervisado (k-NN, MLP, o SVM) como no supervisado (k-means o HMM).

Tradicionalmente las investigaciones optaban en mayor medida por métodos de entrenamiento supervisados, ya que las actividades a clasificar estaban muy acotadas y no tenían gran complejidad. Sin embargo, para obtener un modelo que sea suficientemente generalizador en una aplicación real de RAH hace falta una gran cantidad de datos etiquetados para el entrenamiento, lo cual es un proceso lento y costoso. Por ello, se está optando por clasificadores que utilicen métodos de entrenamiento no supervisado o semi-supervisados, ya que es la manera en la que se pueden obtener grandes

bases de datos con suficiente información como para obtener un modelo general.

Hoy en día las investigaciones en RAH se están centrando en el uso de DNNs para la clasificación de actividades humanas. Esto es debido a que, además de llevar a cabo las tareas de clasificación, es un extractor de características de los datos de entrada tal y como se ha comentado en la sección 1.4.3. Algunos tipos de DNN permiten entrenamiento semi-supervisado de manera que aprenden características de alto nivel de una gran cantidad de datos de entrenamiento no etiquetados, y posteriormente aprende a llevar a cabo las labores de clasificación mediante unos pocos datos etiquetados.

*TABLA II. Clasificadores más utilizados en sistemas de RAH*

| Tipología de clasificador | Clasificador |
|---|---|
| Árboles de decisión | C4.3 y ID3 |
| Bayesianos | Naïve Bayes y Redes Bayesianas |
| Basados en instancias | k-nearest neighbors (k-nn), y k-means |
| Redes neuronales | Support Vector Machines (SVM), Multilayer Perceptron (MLP) y Deep Learning (DNN) |
| Lógica difusa | Fuzzy Basis Function y Fuzzy Inference System |
| Métodos de regresión | Multiple Linear Regresion (MLR), Alternating Logistic Regressions (ALR) |
| Modelos de Markov | Hidden Markov Models (HMM) y Conditional Random Fields (CRF) |
| Conjunto de clasificadores | Boosting y Bagging |

En el caso de aplicaciones en las que no se necesite procesamiento en tiempo real, lo más común es realizar procesamiento *offline*, siendo las redes neuronales basadas en *Deep Learning* los clasificadores más utilizados en la actualidad, tal y como se ha comentado anteriormente [16] [28]. Las técnicas *Deep Learning* son capaces de ir aprendiendo características de complejidad cada vez mayor conforme se va avanzando en sus capas de neuronas, por lo que no es necesario diseñar manualmente conjuntos de características específicos para la aplicación. La clasificación la llevan a cabo en sus últimas capas una vez se han extraído patrones de alto nivel. La gran cantidad de conexiones entre neuronas que presentan este tipo de redes introduce la necesidad de una cantidad muy grande de parámetros a calcular. Para ello es necesario el procesamiento de un gran número de datos de entrenamiento lo que conlleva un alto coste computacional.

### 1.4.5. Métricas de evaluación

Existen distintas métricas para evaluar el correcto funcionamiento de los sistemas de clasificación desarrollados a partir de los resultados obtenidos.

Las métricas más habituales son: matrices de confusión, *accuracy*, *recall*, *F-Measure*, curvas *ROC,* e índice *Kappa* [8] [16].

En el caso de clasificadores binarios, los cuales distinguen si un elemento pertenece a una clase o no, las métricas más representativas son curvas *ROC,* curvas *Precision/Recall,* y matriz de confusión.

Para clasificadores multiclase, los cuales distinguen una de entre varias clases a su salida, la métrica más utilizada es el índice Kappa. El índice Kappa tiene un rango de valores de 0 a 1, siendo un valor superior a 0.7 el indicativo de un buen clasificador [36]. La matriz de confusión también aporta información relevante para distinguir el tipo de error presente entre clases.

En la Tabla III se muestran distintas investigaciones sobre RAH indicando: número de actividades a clasificar, el tipo de sensor utilizado, el modo de extracción de características de los datos (automático o manual), los distintos clasificadores utilizados, y la mejor precisión obtenida con los distintos clasificadores.

*TABLA III. Listado de varios resultados de precisión (accuracy) en RAH*

| Ref | Número actividades | Sensores | Método de extracción de características | Clasificador | Precisión |
|---|---|---|---|---|---|
| [5] | 50 | Acc, Giro | Autom. | CNN | 92.14% |
| [9] | 12 | Acc, Giro | Manual | DA, NB, SVM, HMM, JB, k-NN | 96% (SVM) |
| [7] | 5 | Acc, Giro | Manual | LCSS y SVM | 95% |
| [15] | 7 | Acc, Giro | Autom. | CNN | 99.93% |
| [17] | 6 | Acc, Giro | Autom. | CNN | 98.2% |
| [24] | 12 | Acc | Manual | k-NN, RF, SVM, HMM, k-Means | 98.85% (k-NN) |
| [29] | 17 | Acc | Autom. | CNN+LSTM | 95.8% |
| [31] | 6 | Acc, Giro | Autom. | CNN | 95.75% |
| [33] | 18 | Acc, Giro | Autom., Manual | CNN, SVM, INN, MV, DBN | 96% (CNN) |

## 1.5. **Organización de la memoria**

El resto del proyecto se organiza de la siguiente manera a lo largo de la memoria:

- **Capítulo 2:** En este capítulo se describe el proceso llevado a cabo para la creación de la base de datos. Se comienza explicando el motivo de la elección de los sensores utilizados y su configuración para poder recopilar los datos deseados desde un ordenador. Posteriormente se describe el

modo de lectura de los datos muestreados y el procesamiento realizado. Finalmente se expone el número de muestras que forman la base de datos completa, y la separación realizada de la misma en datos de entrenamiento y datos de test.

- **Capítulo 3:** En el que se explicará los dos tipos de redes implementadas mediante las librerías de TensorFlow para poder llevar a cabo la clasificación de las actividades. La primera red está formada por un modelo convolucional que se ha entrenado con datos no etiquetados, y que lleva a cabo la extracción de características. La segunda red que lleva a cabo la labor de clasificación de las actividades está formada por capas *fully-connected,* utilizando como capa de entrada las salidas obtenidas por la primera red ya entrenada. Al final del capítulo se explican los tres modelos clasificadores desarrollados con estos dos tipos de redes.

- **Capítulo 4:** Donde se exponen y analizan los resultados obtenidos con los distintos modelos clasificadores desarrollados.

- **Capítulo 5:** Donde se exponen las conclusiones obtenidas en el proyecto, y las posibles ampliaciones del trabajo en un futuro.

# Capítulo 2
# Creación de la base de datos

## 2.1. Elección del sensor

Para recopilar información significativa de las actividades se van a utilizar dos nodos sensores, uno colocado en la muñeca del usuario y otro colocado en la cintura. La frecuencia de muestreo de éstos ha de ser de 20 Hz ya que es una frecuencia suficientemente elevada como para poder captar los movimientos realizados por una persona sin conllevar a un consumo elevado de la batería del sensor.

Como ya se había comentado en la sección 1.4.1, los sensores más utilizados en reconocimiento de actividades humanas hasta la actualidad han sido los acelerómetros y giróscopos, ya que la información que proporcionan es suficiente para reconocer un gran número de actividades, y presentan un bajo coste.

Uno de los factores más importantes a la hora de la elección del sensor es el consumo, ya que debido al tipo de aplicación es necesario que la batería del sensor tenga una autonomía de varias horas para poder recopilar muchos datos durante largos periodos de tiempo. Por ello es necesario un dispositivo que tenga una tecnología inalámbrica de bajo consumo, siendo *Bluetooth Low Energy* (BLE) la más utilizada para las frecuencias con las que trabajan los acelerómetros y giróscopos.

*TABLA IV. Distintos dispositivos portables comparados*

| Fabricante | Modelo | Tecnología inalámbrica | Precio |
|---|---|---|---|
| Texas Instruments | CC2650STK | Bluetooth Low Energy | 25,01 € |
| Dialog Semiconductors | SmartBond DA14583 | Bluetooth Low Energy | 38,30 € |
| mbientlab | Wristband sensor research kit | Bluetooth Low Energy | 83,50€ |

En la Tabla IV se muestran algunas de las opciones encontradas en el mercado que cumplen con los requisitos expuestos. De acuerdo con las especificaciones comentadas en el párrafo anterior, en este proyecto se ha optado por utilizar el dispositivo SensorTag CC2650STK de Texas

Instruments. Esta elección es debida a su reducido precio, a que utiliza la tecnología inalámbrica de bajo consumo BLE con la que la batería del dispositivo puede llegar a durar de semanas a años dependiendo de la frecuencia de envío de datos, y a que es de capaz de reprogramarse mediante un *debugger* para obtener las frecuencias de muestreo deseadas. Además, Texas Instruments proporciona una página de soporte a usuarios (*TI E2E Community*), y una plataforma de iniciación para desarrolladores (*SimpleLink Academy*) donde se pueden encontrar distintos tutoriales para la comprensión de la tecnología BLE integrada dentro del dispositivo, o tutoriales de programación de éste. En la Fig. 2.1 se pueden observar los sensores SensorTag junto con las fundas utilizadas para fijarlos a la muñeca y a un cinturón.

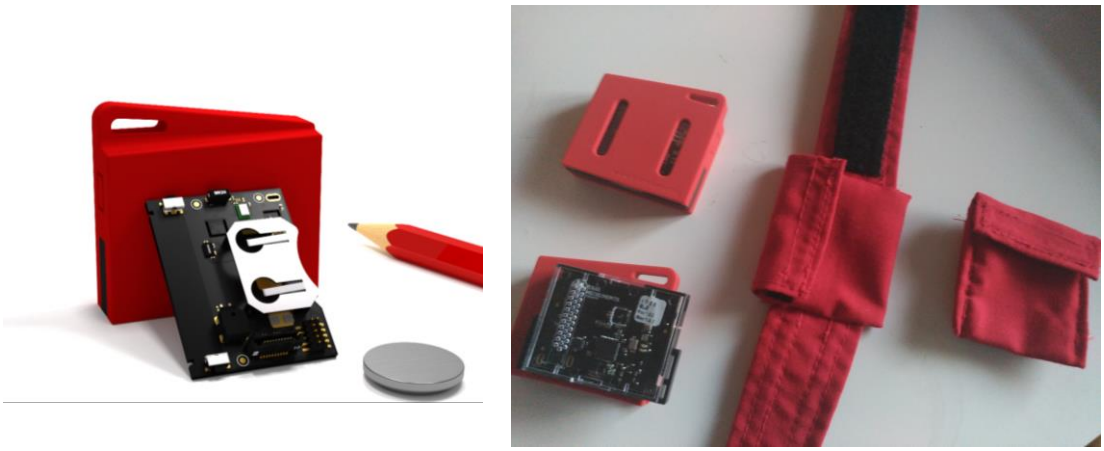


*Fig. 2.1. Imágenes de los SensorTag utilizados y de las fijaciones*

### 2.1.1. Bluetooth Low Energy en SensorTag CC2650

Como ya se ha comentado, el *Bluetooth Low Energy* (BLE) es un protocolo de comunicación inalámbrica de bajo consumo que permite comunicar una gran variedad de dispositivos. Al estar diseñado para que tenga un consumo bajo de energía, es común encontrar este protocolo en la mayoría de los dispositivos portables.

En la Fig. 2.2 se puede observar un esquema de las capas que forman el protocolo BLE, obtenido de la guía de diseño BLE de Texas Instruments [37], donde se destacan las capas de mayor nivel *Generic Access Profile* (GAP) y *Generic Attribute Pofile* (GATT). La capa GAP es la encargada de hacer que un dispositivo BLE sea visible por el resto para poder realizar la conexión entre ellos. La capa GATT es la encargada de la comunicación bidireccional entre dos dispositivos de manera que permite la transferencia de información entre ambos.

De esta manera, en una comunicación por BLE los dispositivos pueden tener distintos roles en cada una de estas capas. En la capa GAP un dispositivo puede tomar la función de dispositivo esclavo cuando éste advierte al resto de dispositivos BLE de la posibilidad de establecer una conexión con él, o puede tomar la función de dispositivo maestro si es él quien efectúa la solicitud de conexión con otros dispositivos esclavos visibles. En la capa GATT existen los roles de servidor y cliente dependientes del

papel que efectúan en la transmisión de información. El servidor GATT será aquel dispositivo que contiene la información que se envía, mientras que el cliente GATT es el dispositivo que recibe dicha información. De esta manera, tanto un dispositivo maestro como uno esclavo puede tomar el papel de servidor o cliente.

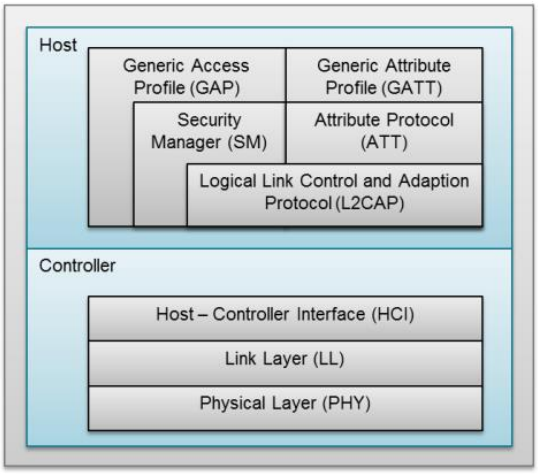



*Fig. 2.2. Arquitectura del protocolo Bluetooth Low Energy*

Para poder realizar solicitudes y envío de información o instrucciones entre dos dispositivos BLE, la capa GATT dispone de unos recursos denominados servicios y características. Las características GATT son los valores que se desean leer o escribir sobre un dispositivo, como por ejemplo el valor de las medidas del sensor de movimiento. Estas características también tienen unos descriptores que indican información acerca de la característica. Los servicios GATT son conjuntos de características agrupadas habitualmente por bloques funcionales. Como ejemplo, un servicio englobaría la característica del valor del sensor de movimiento y otras características para la configuración de dicho sensor.

En la comunicación BLE, el dispositivo que actúa como cliente GATT puede solicitar leer el valor de una determinada característica al dispositivo servidor de manera que éste se la envíe. Sin embargo, este protocolo también permite el envío continuo del valor de una característica cada vez que se actualice, sin necesidad de que el cliente GATT haga una solicitud de lectura de la característica al servidor. Este tipo de lectura de datos se denomina notificaciones y se habilitan por medio de una característica GATT que el cliente modifica en el dispositivo servidor. Un ejemplo sería activar las notificaciones de un servicio GATT correspondiente a un sensor incorporado en un dispositivo servidor para que se envíe automáticamente cada valor sensado al cliente GATT al que esté conectado.

Tanto los servicios como las características son accesibles mediante un identificador único UUID de 128 bits proporcionado por la capa *Attribute Protocol* (ATT). Todas las características y servicios del SensorTag tienen un identificador base en el que solo cambian 16 bits entre ellos, siendo éste F000XXXX-0451-4000-B000-000000000000, y donde las X representan a los 16 bits que varían. Además, Texas Instruments proporciona un identificador personalizado a cada característica del SensorTag para así poder acceder a ellas más fácilmente con las API desarrolladas por ellos.

## 2.2. **Configuración de los sensores**

Como ya se ha comentado, en este proyecto se han utilizado dos sensores SensorTag CC2650STK, uno colocado en una muñeca y otro en la cintura. Para poder configurar y recopilar los datos muestreados por los dos sensores al mismo tiempo desde un ordenador, se ha hecho uso del CC2540 USB Dongle de Texas Instruments (Fig. 2.3) que funciona como interfaz entre el protocolo *Bluetooth Low Energy* y un puerto serie. El control de este adaptador se lleva a cabo mediante un software proporcionado por Texas Instruments llamado BTool.

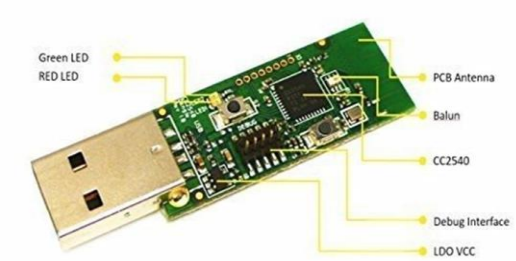



*Fig. 2.3. Dispositivo CC2540 USB Dongle de Texas Instruments*

Este software permite al dispositivo USB actuar como maestro GAP en la comunicación mientras que los SensorTag funcionan como dispositivos esclavos. En sistemas con sensores portables, como el de este proyecto, los sensores (dispositivos esclavos) actúan como servidores GATT, ya que son los que captan la información, mientras que el USB dongle (dispositivo maestro) actúa como cliente en la transmisión de datos por BLE, ya que es el que capta los datos solicitados al servidor.

La aplicación BTool permite llevar a cabo distintas acciones en la comunicación, como seleccionar los parámetros de conexión deseados, o descubrir, leer y escribir características GATT de los dispositivos que actúan como servidores. La información acerca de los paquetes de datos recibidos o enviados queda almacenada en un *Log* que se puede guardar como fichero de texto.

### 2.2.1. **Configuración del sensor de movimiento de los SensorTag**

Los SensorTag llevan incorporados un sensor de movimiento MPU9250, el cual proporciona información en los tres ejes de acelerómetro, giróscopo y magnetómetro. Para configurar dicho sensor, el *firmware* del SensorTag dispone de características GATT para ello. Esto permite que el dispositivo maestro pueda habilitar o deshabilitar cada sensor, cambiar el periodo de muestreo de estos (con ciertas restricciones), o cambiar el rango de medida del acelerómetro. Para ello se pueden configurar las características BLE disponibles del SensorTag a través de la aplicación BTool, sin necesidad de reprogramarlo con el *debugger*.

En la Fig. 2.4, obtenida de la guía de usuario del SensorTag [38], se pueden observar las características que permiten leer o configurar el sensor de movimiento (están agrupadas en un servicio). En ella se indica la UUID de las características que forman el servicio del sensor de movimiento, marcándolas con un asterisco en caso de que sea una UUID reducida (solo los 16 bits que cambian).

| Type | UUID | Access | Size (bytes) | Description |
|---|---|---|---|---|
| Data | AA81* | R/N | 18 | GyroX[0:7], GyroX[8:15], GyroY[0:7], GyroY[8:15], GyroZ[0:7], GyroZ[8:15], AccX[0:7], AccX[8:15], AccY[0:7], AccY[8:15], AccZ[0:7], AccZ[8:15], MagX[0:7], MagX[8:15], MagY[0:7], MagY[8:15], MagZ[0:7], MagZ[8:15] |
| Notification | 2902 | R/W | 2 | Write 0x0001 to enable notifications, 0x0000 to disable. |
| Configuration | AA82* | R/W | 2 | One bit for each gyro and accelerometer axis (6), magnetometer (1), wake-on-motion enable (1), accelerometer range (2). Write any bit combination top enable the desired features. Writing 0x0000 powers the unit off. |
| Period | AA83* | R/W | 1 | Resolution 10 ms. Range 100 ms (0x0A) to 2.55 sec (0xFF). Default 1 second (0x64). |

*Fig. 2.4. Características BLE del servicio del sensor de movimiento*

Con la aplicación BTool se puede descubrir todas las características GATT de los dispositivos servidores conectados al USB dongle (dispositivo cliente). Buscando las UUID de la Fig. 2.4 en las características descubiertas, podemos obtener cuales son los *handles* que utiliza Texas Instruments para leer o escribir características GATT, tal y como se muestra en la Fig. 2.5. También se pueden ver en color rojo y naranja los descriptores del servicio y características respectivamente.

A continuación, se muestra los valores a escribir en las características GATT para poder configurar cada SensorTag con un periodo de muestreo de 50 ms (20 Hz), con notificaciones habilitadas, y con los sensores y rangos deseados.

**Configuración del periodo de muestreo**

El firmware incorporado por defecto en el SensorTag permite configurar los periodos de muestreo mediante características GATT únicamente en el rango de 100 milisegundos (ms) a 2.55 segundos, tal y como indica en la Fig. 2.4. Puesto que se necesita obtener al menos una frecuencia de muestreo de 20 Hz (50 ms) ha sido necesario modificar el código original de los SensorTag utilizando para ello el *debugger* DevPack de Texas Instruments [39]. De este modo se ha modificado el límite inferior del rango para que se reduzca a 50 ms, manteniendo la resolución de 10 ms indicada en la Fig. 2.4.

| Handle | Uuid | Uuid Description |
|---|---|---|
| 0x003A | 0x2800 | GATT Primary Service Declaration |
| 0x003B | 0x2803 | GATT Characteristic Declaration |
| 0x003C | 0xF000AA8104514000B000000000000000 | Unknown |
| 0x003D | 0x2902 | Client Characteristic Configuration |
| 0x003E | 0x2803 | GATT Characteristic Declaration |
| 0x003F | 0xF000AA8204514000B000000000000000 | Unknown |
| 0x0040 | 0x2803 | GATT Characteristic Declaration |
| 0x0041 | 0xF000AA8304514000B000000000000000 | Unknown |

*Fig. 2.5. Características del sensor de movimiento en BTool*

Para configurar el periodo de muestreo del sensor de movimiento a 50 ms con características GATT, hay que escribir el valor hexadecimal "0x0005" en la característica con *handle* "0x0041".

**Habilitación de notificaciones**

La habilitación de las notificaciones de las lecturas del sensor de movimiento se realiza escribiendo "0x0001" en la característica con *handle* "0x003D" tal y como se muestra en las figuras 2.4 y 2.5. De este modo cada vez que se obtenga una muestra se enviará su valor directamente al dispositivo cliente GATT.

**Habilitación y configuración del acelerómetro y giróscopo**

En la Fig. 2.6 se muestra de forma más detallada el modo de configuración del sensor de movimiento mediante la característica GATT correspondiente.

Puesto que el sensor de movimiento está compuesto por un acelerómetro, un giróscopo, y un magnetómetro, hay que habilitar solo los sensores deseados. Tal y como se describe en la Fig. 2.6 se habilita el acelerómetro y giróscopo escribiendo un 1 en los 6 bits menos significativos (0 al 5) de la característica con *handle* "0x003F". El magnetómetro se deja inhabilitado escribiendo un 0 en el bit 6 del registro de la característica.

| Bits | Usage |
|---|---|
| 0 | Gyroscope z axis enable |
| 1 | Gyroscope y axis enable |
| 2 | Gyroscope x axis enable |
| 3 | Accelerometer z axis enable |
| 4 | Accelerometer y axis enable |
| 5 | Accelerometer x axis enable |
| 6 | Magnetometer enable (all axes) |
| 7 | Wake-On-Motion Enable |
| 8:9 | Accelerometer range (0=2G, 1=4G, 2=8G, 3=16G) |
| 10:15 | Not used |

*Fig. 2.6. Uso de característica de configuración del sensor de movimiento*

La elección del rango del acelerómetro se realiza mediante los bits 8 y 9 del valor de la característica de configuración. Puesto que la aplicación es para detección de actividades humanas que pueden conllevar aceleraciones altas, se escoge la escala más alta de 16 *G*. De este modo, se escribirá sobre la característica GATT con *handle* "0x003F" el valor "0x033F" para habilitar los sensores deseados, y seleccionar el rango del sensor indicado.

Como se puede observar, el código original en el SensorTag no permite modificar el rango del giróscopo mediante características GATT. Su valor está configurado por defecto en $\pm 250\ deg/s$ tal y como se describe en la guía de usuario del SensorTag [38]. Debido a que este rango tiene límites demasiado bajos para los valores que se alcanzarán en el desempeño de actividades humanas, los datos obtenidos saturarían en caso de dejar configurado ese rango del sensor.

Buscando en el registro 27 del datasheet del sensor de movimiento MPU9250 [40], se puede comprobar como el rango del giróscopo puede configurarse con los valores de $\pm 250, \pm 500, \pm 1000, y\ \pm 2000\ deg/s$. Para

configurarlo, se modifica el código del SensorTag para que se escale el rango del giróscopo a $\pm 2000\ deg/s$ mediante las librerías que dispone para escribir sobre los registros del MPU9250.

## 2.3. Lectura y procesamiento de datos

### 2.3.1. Lectura de los datos obtenidos en el Log

Tras conectar los dos SensorTag al dispositivo *USB dongle* mediante la aplicación BTool, y configurarlos según se ha descrito en el apartado anterior, se reciben las notificaciones de los datos muestreados por los dos SensorTag y se almacenan en formato de Log, tal y como se observa en la Fig. 2.7. Como se puede apreciar en la figura, cada mensaje recibido en el Log viene con:

- El *timestamp* en el que se ha recibido.
- El *handle* de conexión que indica la identificación del esclavo del que se ha recibido el mensaje.
- El handle de la característica que corresponde con la que se indicaba en las figuras 2.4 y 2.5 como la que contiene los datos del sensor.
- El valor de dicha característica el cual contiene los valores de cada eje de acelerómetro y giróscopo en formato hexadecimal *little-endian* de 16 bits, tal y como indicaba la Fig. 2.4.

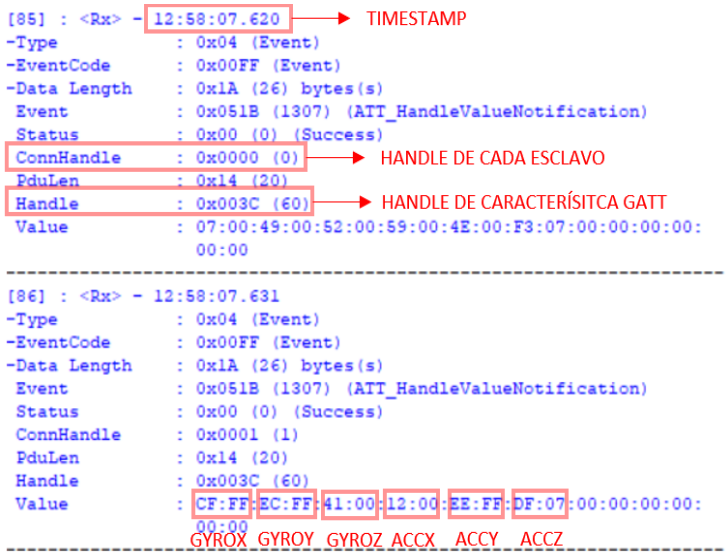



*Fig. 2.7. Ejemplo de datos enviados por los sensores, en el Log de BTool*

Una vez guardado en fichero de texto el Log obtenido durante el desempeño de actividades, se crea una tabla con todos los datos recibidos mediante el tratamiento del fichero en Matlab con un programa creado para dicha tarea.

Este programa introduce en cada fila de una tabla los datos correspondientes a las medidas seguidas de los dos SensorTag. Es decir, aunque las medidas tomadas por los dos SensorTag no pueden ser sincronizadas, como el

tiempo que hay entre ellas es de unos 10 ms se asume que se han tomado al mismo tiempo. De este modo, en cada medida registrada en la base de datos habrá 12 valores correspondientes a los 3 ejes del acelerómetro y 3 ejes de giróscopo de los dos SensorTag. Antes de introducirlos en la tabla los datos, hay que convertirlos de hexadecimal de 16 bits a las unidades deseadas. Como ejemplo, se muestra la transformación de los datos del acelerómetro a unidades de G en la Fig. 2.8.

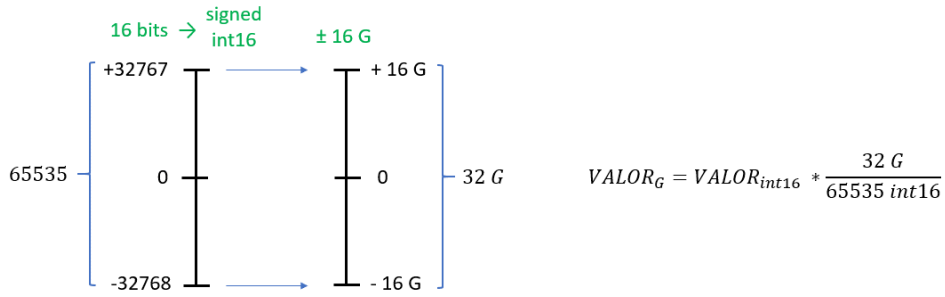



*Fig. 2.8. Transformación de unidades de los datos del acelerómetro*

En caso de que en el fichero se encuentren dos medidas seguidas del mismo SensorTag (indicado por su *handle* de conexión), se asume la pérdida del dato de uno de los sensores, y se realiza una interpolación de los datos anterior y siguiente al valor perdido para aproximar el valor real.

Para cada medida de los sensores introducida en la tabla, se le introduce la etiqueta de la actividad realizada. De este modo se podrán aplicar posteriormente las técnicas de *machine learning* con las que obtener un modelo clasificador de las actividades etiquetadas. También se introduce la etiqueta del usuario que ha realizado la actividad para así poder comprobar de qué sujeto es la medida tomada, o comprobar si el modelo generaliza bien para todos los usuarios.

Una vez obtenidos todos los datos del Log en una tabla, se transforma a formato *csv* para que sea fácilmente leíble por las librerías de cualquier lenguaje. En este caso, se utilizan las librerías de Pandas en Python para leer estos archivos con extensión *csv*. De este modo se crea una base de datos *raw data* con las medidas de los sensores recopiladas durante el desempeño de todas las actividades por cada usuario.

La base de datos creada está formada por un pequeño conjunto de datos etiquetados, y una gran cantidad de datos sin etiquetar. El conjunto de datos no etiquetados se utilizará para realizar aprendizaje no supervisado con el que se consigue que el conjunto de datos etiquetados sea mucho menor. Los datos no etiquetados han sido recopilados durante periodos de una hora por sujeto, en los cuales se han llevado a cabo actividades aleatorias, o similares a las del conjunto de datos etiquetados. Los datos etiquetados constan de 7 actividades cotidianas y 4 golpes de tenis. El tiempo de recopilación de datos de cada una de las actividades etiquetadas ha sido de 5 minutos por sujeto.

Cada actividad está realizada por 8 sujetos distintos que presentan variabilidad en sus dimensiones físicas, peso, y sexo, tal y como se observa en la Tabla V. Las distintas características de los sujetos nos ayudarán a comprobar la robustez del modelo.

*TABLA V. Características de los sujetos que realizan la base de datos*

| Sujeto | Sexo | Edad | Altura (m) | Peso (kg) | Lateralidad |
|---|---|---|---|---|---|
| 1 | M | 23 | 1,74 | 73 | Diestro |
| 2 | M | 24 | 1,90 | 80 | Diestro |
| 3 | M | 21 | 1,82 | 76 | Diestro |
| 4 | M | 30 | 1,68 | 70 | Diestro |
| 5 | F | 23 | 1,60 | 65 | Zurdo |
| 6 | F | 27 | 1,64 | 59 | Diestro |
| 7 | F | 23 | 1,59 | 57 | Diestro |
| 8 | F | 19 | 1,69 | 62 | Diestro |

### 2.3.2. Procesamiento de los datos

Para conseguir llevar a cabo el objetivo de este proyecto el cual implica obtener un modelo clasificador de actividades general para cualquier usuario, se ha llevado a cabo el procesamiento y segmentación de los datos de los sensores mediante el cálculo de los espectrogramas de la señal.

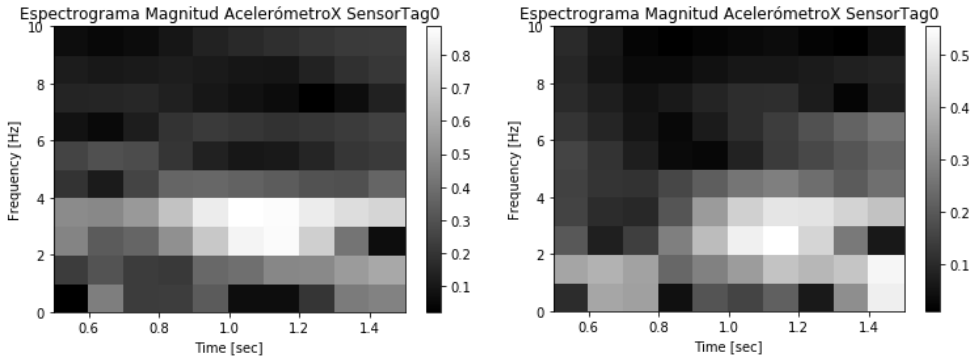



*Fig. 2.9. Ejemplos de espectrograma de actividad "correr" realizada por dos usuarios de distinto sexo y altura*

Al tratar con espectrogramas en lugar de los datos en crudo de los sensores como entrada a un clasificador, se pretende conseguir que las diferencias en las características de cada usuario (edad, peso, altura, longitud de extremidades, fuerza…) no tengan influencia sobre los resultados obtenidos. Esto es debido a que, aunque distintos usuarios realicen una misma actividad a distintas frecuencias, la forma de la evolución temporal de las frecuencias observada en los espectrogramas será la misma durante el desempeño de la actividad. Es decir, el clasificador será capaz de distinguir las formas características de los espectrogramas de una actividad determinada independientemente de si está desplazada/estirada en frecuencias o en tiempo. En la Fig. 2.9 se observa un ejemplo del espectrograma obtenido por dos sujetos distintos corriendo. Como se puede comprobar, ambos presentan formas similares aunque las magnitudes de la potencia registradas por un sujeto sean mayores que las del otro.

La segmentación de la señal necesaria para el cálculo de los espectrogramas se lleva a cabo mediante una ventana deslizante a lo largo de todas las señales. Hay que tener en cuenta que la señal a segmentar ha de ser de una única actividad realizada por un usuario, ya que si no fuera así, no se podría asignar una etiqueta de actividad al espectrograma. De este modo, se realiza la segmentación en paralelo de las 12 señales provenientes de cada eje de los sensores, con los siguientes parámetros de las ventanas deslizantes:

- La longitud de cada ventana es de 20 muestras, lo cual corresponde a 1 segundo de recopilación de datos (con la frecuencia de muestreo de 20 Hz).
- La superposición entre ventanas es del 90% (18 muestras superpuestas)
- En las ventanas se utiliza el enventanado de Hann para evitar que la transformada de Fourier discreta calculada en cada ventana del espectrograma tenga en cuenta las discontinuidades producidas en los bordes de la ventana (*aliasing*).

Teniendo en cuenta las características seleccionadas de la segmentación de las señales, se seleccionan los siguientes parámetros para el cálculo de los espectrogramas:

- El número de muestras de la señal que abarca un espectrograma son 40, que es el equivalente a 2 segundos de recopilación de datos de los sensores.
- La superposición de las muestras tomadas para calcular un espectrograma respecto del anterior es del 90% (36 muestras superpuestas)
- La resolución frecuencial de la transformada de Fourier calculada para cada ventana está fijada en la mitad de la frecuencia de muestreo de los sensores (10 Hz) para así cumplir con el teorema de Nyquist. De este modo, se calculan 11 valores de la transformada de Fourier discreta, contando con el valor calculado para 0 Hz.
- La resolución temporal es igual al número de ventanas obtenidas, que depende de los parámetros escogidos: número de muestras por espectrograma, número de muestras por ventana, y superposición entre ventanas. En este caso el número de ventanas es igual a 11.

Con la selección de los parámetros descritos se obtienen espectrogramas de dimensión 11x11. En la Fig. 2.9 se muestran dos ejemplos de los espectrogramas obtenidos, visualizándolos en dos dimensiones, donde la tercera dimensión representada en tono de grises corresponde a la magnitud calculada en la transformada de Fourier discreta. En dicha representación cada "pixel" del espectrograma es el promedio de dos datos contiguos (por lo que se ha representado como imágenes 10x10).

# 2.4. Base de datos

## 2.4.1. Base de datos completa

En la Tabla VI se refleja el número de muestras que hay por cada eje de sensor en la base de datos completa. En la tabla se encuentra información tanto del número de muestras *raw data* recopiladas por cada eje de los sensores, como del número de espectrogramas obtenidos a partir de esos datos. Se ha representado el número de datos por eje, ya que cada uno de los 12 ejes (3 acelerómetro y 3 giróscopo de cada SensorTag) corresponde a la medida en un mismo instante temporal, por lo que luego se introducirán conjuntamente al modelo clasificador.

*TABLA VI. Nº de muestras de base de datos completa por cada eje de sensor.*

| **Actividad** | **Nº muestras RawData por eje** | **Nº espectrogramas por eje** |
|---|---|---|
| No etiquetada | 577.097 | 144.201 |
| Andar | 49.791 | 12.373 |
| Correr | 42.679 | 10.605 |
| Saltar | 48.387 | 12.022 |
| Agacharse/Incorporarse | 48702 | 12.100 |
| Estar de pie | 49.845 | 12.386 |
| Estar sentado | 50.847 | 12.637 |
| Sentarse/Levantarse | 48.654 | 12.089 |
| Golpe Drive | 30.085 | 7.475 |
| Golpe Revés | 30.203 | 7.504 |
| Golpe Mate | 30.261 | 7.518 |
| Golpe Globo | 30.135 | 7.488 |
| | **1.036.686** | **258.398** |

## 2.4.2. Separación de la base de datos etiquetados

Con el fin de poder evaluar el modelo clasificador desarrollado con datos que no hayan sido aprendidos por el mismo, se realiza una separación de la base de datos de espectrogramas de las actividades etiquetadas. De este modo, se ha obtenido el conjunto para entrenamiento y el conjunto para test. Puesto que en este proyecto uno de los objetivos principales es conseguir obtener un modelo robusto ante la variabilidad de los posibles sujetos que realicen las actividades, se han extraído dos conjuntos de test distintos.

En el primer conjunto de test se coge una parte de los datos de 7 de los 8 usuarios que han realizado las actividades. Para ello, por cada usuario se extraen los 10 segundos de recopilación de datos que están situados en medio de la secuencia temporal de cada actividad. Estos 10 segundos son correspondientes a 200 muestras *raw data* con las que se extraen 41 espectrogramas por actividad de cada usuario (teniendo en cuenta la superposición entre espectrogramas indicada en la sección 2.3.2). De este modo, se obtiene un conjunto de test de datos no vistos durante el entrenamiento, pero que pertenecen a las mismas personas que recopilaron los datos de entrenamiento.

El segundo conjunto de test está formado por todos los datos recopilados de cada actividad por el usuario que no había sido contemplado en el conjunto los conjuntos de entrenamiento y test en el caso anterior. De este modo, se obtienen unos datos que no han sido vistos durante el entrenamiento, ni han sido recopilados por la misma persona que realizó el conjunto de datos de entrenamiento. Con este conjunto se comprueba cual sería la eficacia del modelo desarrollado en una situación real, de manera que se observará su capacidad generalizadora. El usuario seleccionado ha sido el usuario 1 de la Tabla V ya que como se puede observar sería el usuario medio de entre los que han recopilado los datos. Al utilizar el usuario intermedio se pretende que los resultados obtenidos sean referentes a los que se obtendrían con la mayoría de las personas en una aplicación real.

En la Tabla VII se ha reflejado número de espectrogramas que contiene el conjunto de entrenamiento y los dos conjuntos de test.

*TABLA VII. Nº de espectrogramas de cada separación realizada de la base de datos etiquetada completa*

| **Actividad** | **Nº de datos de entrenamiento** | **Nº de datos de test de usuarios ya vistos por el modelo** | **Nº de datos de test de usuario no visto por el modelo** |
|---|---|---|---|
| Andar | 10.552 | 287 | 1.534 |
| Correr | 8.807 | 246 | 1.552 |
| Saltar | 10.300 | 287 | 1.435 |
| Agacharse/Incorporarse | 10.272 | 287 | 1.541 |
| Estar de pie | 10.573 | 287 | 1.526 |
| Estar sentado | 10.772 | 287 | 1.578 |
| Sentarse/Levantarse | 10.307 | 287 | 1.495 |
| Golpe Drive | 5.802 | 164 | 1.509 |
| Golpe Revés | 5.832 | 164 | 1.508 |
| Golpe Mate | 5.834 | 164 | 1.520 |
| Golpe Globo | 5.823 | 164 | 1.501 |
| | **94.874** | **2.624** | **16.699** |

# Capítulo 3
# Modelo clasificador

Para llevar a cabo la clasificación de las actividades deseadas con la información proporcionada por los espectrogramas, se han implementado modelos clasificadores con aprendizaje semi-supervisado.

Este tipo de aprendizaje es muy útil ya que permite que un modelo aprenda características representativas de un conjunto de datos no etiquetados, y luego las utilice para aprender a clasificar datos del mismo tipo que sí están etiquetados. De esta manera se consigue reducir considerablemente la cantidad de datos etiquetados necesaria, utilizando en su lugar una gran cantidad de datos no etiquetados los cuales conllevan mucho menor esfuerzo de recopilación.

Como se ha comentado, el aprendizaje semi-supervisado consta de dos fases. En la primera fase, se construye un modelo de aprendizaje no supervisado que sea capaz de aprender características representativas de los datos no etiquetados. Es conveniente que estos datos no etiquetados sean del mismo tipo que los etiquetados para que las características aprendidas sean representativas de los datos que se quieren clasificar. En la segunda fase, se utiliza el modelo entrenado en la primera fase para obtener las características aprendidas sobre los datos etiquetados, de manera que con ellas se entrene un modelo clasificador de manera supervisada.

Como modelo de aprendizaje no supervisado se ha seleccionado el Autoencoder convolucional. Una vez entrenado el Autoencoder utilizando indistintamente los espectrogramas de cualquier sensor y actividad, se utilizarán las primeras capas entrenadas (*encoder*) para codificar cada una de las 12 señales provenientes de los dos sensores. Concatenando las 12 codificaciones de los sensores se dispone de la información de entrada a un perceptrón multicapa (MLP). El MLP será entrenado de manera supervisada para llevar a cabo las clasificaciones de las actividades etiquetadas.

## 3.1. **Autoencoder convolucional**

Los Autoencoders son un tipo de red neuronal artificial capaces de aprender representaciones eficientes de los datos de entrada, llamadas codificaciones, sin utilizar datos etiquetados. Por ello, estas redes pueden usarse como potentes extractores de características de los datos de entrada [41].

Este tipo de redes se entrenan aprendiendo a replicar sus entradas en sus salidas. La clave de su funcionamiento es que los datos de entrada se comprimen en las primeras capas ocultas (antes de la capa de intermedia) para volverse a descomprimir en las ultimas capas ocultas hasta su salida (después de la capa intermedia). De esta manera se suele decir que los Autoencoders están compuestos por una primera parte llamada *encoder*, y una segunda parte llamada *decoder*. En caso de que el autoencoder tenga más de una capa oculta, se le suele denominar *Stacked Autoencoder* o *Deep Autoencoder*. En la Fig. 1.5 del capítulo 1 puede observarse la estructura que presentan este tipo de redes.

Una vez entrenado el Autoencoder, se puede utilizar la parte *encoder* como extractor de características para alimentar al MLP (capas *fully-connected*) entrenado con datos etiquetados de clasificación.

En este proyecto se ha utilizado una variante de este tipo de redes, la cual se llama Autoencoder convolucional. Su estructura es la misma que el Autoencoder, pero cambiando las capas *fully-connected* (como las mostradas en la Fig. 1.5) por capas convolucionales. De esta manera, tratando los espectrogramas como imágenes, se consigue que las características encontradas en ellos sean reconocibles independientemente de la escala y posición de las mismas. Gracias a esta invariancia, es posible conseguir que el modelo generalice a cualquier usuario que realice las actividades, ya que no importará si las formas obtenidas en los espectrogramas están desplazadas/estiradas en frecuencia o tiempo, tal y como se ha comentado en la sección 2.3.2.

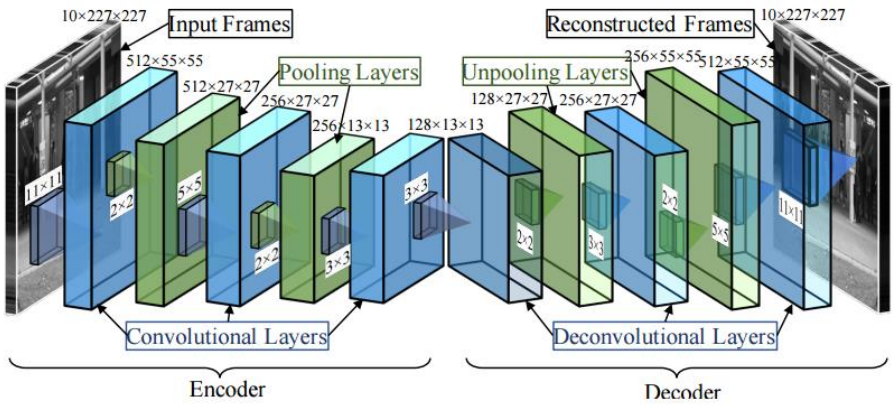



*Fig. 3.1. Estructura de un Autoencoder convolucional*

En la Fig. 3.1 [42] se muestra un ejemplo de la estructura típica de un Autoencoder convolucional. Tras cada capa convolucional de la parte *encoder* de la red se van obteniendo mapas de características de mayor nivel de abstracción. En la parte *decoder* de la red se aplican capas deconvolucionales para así intentar reconstruir los datos de entrada a partir de los mapas de características obtenidos.

Para el entrenamiento del Autoencoder se utilizarán los espectrogramas obtenidos con los datos no etiquetados. Tal y como se ha comentado al final de la sección 2.3.1, el conjunto de datos no etiquetados está formado por 144.201 espectrogramas. Dado que se entrena un único autoencoder con todos los espectrogramas indistintamente de la señal de la que procedan, se

consigue configurar un módulo codificador universal de la señal que será replicado para las 12 señales.

### 3.1.1. Parámetros de diseño de la arquitectura

El Autoencoder convolucional construido para llevar a cabo la extracción de características de los espectrogramas está formado por tres capas convolucionales y tres capas deconvolucionales, tal y como se ha representado en la Fig. 3.2.

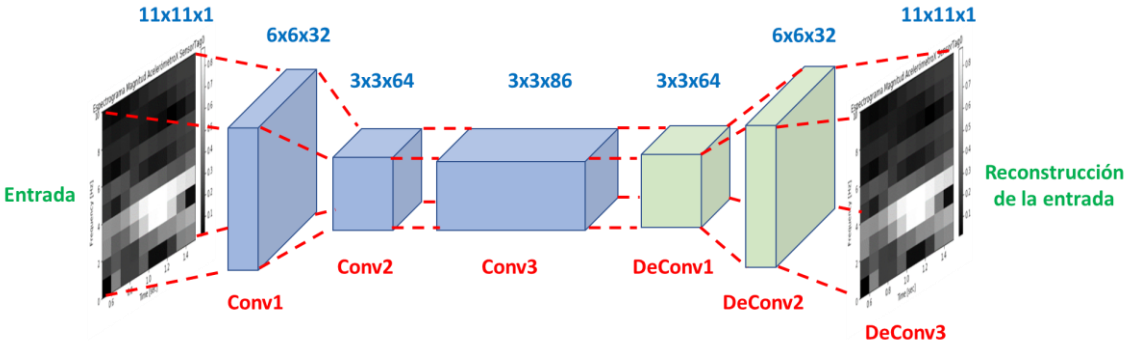



*Fig. 3.2. Arquitectura del Autoencoder convolucional utilizado*

Los filtros (o *kernels* de convolución) de cada una de las capas convolucionales son de 2x2. Estos filtros son los pesos que se irán ajustando durante el entrenamiento para llevar a cabo el reconocimiento de patrones de los espectrogramas. Cada mapa de características de una capa convolucional comparte los mismos pesos de conexión a lo largo de todo el mapa (utiliza el mismo filtro a lo largo de toda la imagen de entrada de la capa). Este filtro es desplazado a lo largo y ancho de la imagen hasta convolucionarla completamente. Este valor de desplazamiento se denomina *stride* y suele ser igual para ambas direcciones de desplazamiento.

Debido a la baja dimensionalidad de los espectrogramas (11x11) no se utilizan capas de *pooling*, ya que introducirían una fuerte pérdida de información. Las capas de *pooling* se diseñan para reducir considerablemente la dimensionalidad de la respuesta de salida de la capa convolucional, pero se aplican principalmente en imágenes con mucha más resolución. En su lugar, se va reduciendo la dimensionalidad de los mapas de características obtenidos aplicando *strides (*con valor mayor que uno) del filtro aplicado sobre la entrada de la capa. Tal y como se observa en la Fig. 3.2, en las capas Conv1, Conv2, Deconv2 y Deconv3 se ha utilizado *stride* de dos por lo que la dimensión de los mapas de características se reduce a la mitad en el caso de la convolución, y se duplica en el caso de la deconvolución. Las capas Conv3 y Deconv1 tienen un *stride* de 1 por lo que no se modifica la dimensión de los mapas de características obtenidos. El número de mapas (equivalentes a canales de color filtrado) que se obtienen en cada capa se ha representado en la Fig. 3.2 como la tercera dimensión de las capas.

Como función de activación de las capas convolucionales se escoge una de las funciones no saturables, siendo la función de activación ELU (*Exponential Linear Unit*) la que mejores resultados suele dar [41].

Los pesos de los mapas de características de cada capa convolucional son inicializados según la estrategia de inicialización *He initialization* ya que es la que mejor suele funcionar con la función de activación ELU [41].

La función de coste utilizada es el error cuadrático medio (*Mean Squared error*, MSE), calculado mediante la diferencia entre la entrada introducida que queremos replicar y la reconstrucción obtenida a la salida de la red.

El algoritmo de entrenamiento utilizado para ajustar los pesos de la red es el descenso por el gradiente junto con la técnica de optimización *Adam,* ya que esta técnica acelera el entrenamiento de la red al ser mucho más rápido que el descenso por el gradiente habitual [41].

### 3.1.2. Parámetros del entrenamiento

Para entrenar el modelo con el algoritmo de entrenamiento indicado anteriormente, se ha seleccionado un factor de entrenamiento de 0.0001 con el cual el algoritmo llega a converger en un número de ciclos (*epochs*) razonable.

Para mejorar los resultados obtenidos, y acelerar el entrenamiento, se ha realizado entrenamiento por *mini-batch,* de manera que el conjunto de datos de entrenamiento se ha ido introduciendo a la red por lotes, en vez de todo el conjunto al mismo tiempo. Cada vez que se introduce un lote a la red, se calcula el error obtenido y se ajustan sus pesos con el error promedio obtenido del lote. De este modo, el descenso por el gradiente es más eficiente al ir calculando el gradiente poco a poco, sin excesivas oscilaciones. El tamaño de los lotes seleccionado ha sido de 96 espectrogramas (8 de cada uno de los 12 ejes de los sensores).

Antes de introducir cada lote de datos a la capa de entrada de la red, se normalizan con la media y desviación estándar de todos los datos de entrenamiento. De este modo se consigue centrar la distribución de los datos de entrenamiento con lo que el modelo aprende las escala óptima y media de las entradas introducidas a la capa de entrada. Este proceso mejora los resultados obtenidos, y consigue que el modelo sea menos sensible a la inicialización de los pesos.

El número de ciclos ejecutados durante el entrenamiento es de 100, ya que es el valor a partir del cual el error MSE obtenido deja de mejorar y alcanza una zona plana.

## 3.2. Perceptrón multicapa MLP

Tras entrenar el Autoencoder de manera no supervisada, se consigue obtener un extractor de características de los datos de entrada utilizando sus capas convolucionales (representadas en azul en la Fig. 3.2). Este extractor es replicado doce veces (una por eje de sensor), de manera que las características obtenidas de cada uno de ellos son introducidas conjuntamente a un clasificador supervisado.

El clasificador utilizado es el MLP. No es de interés la aplicación de clasificadores *Deep* ya que su uso está pensado para llevar a cabo tanto la labor de extracción de características como de clasificación. En nuestro caso, la extracción de características se ha conseguido utilizando un modelo entrenado de manera no supervisada, por lo que un simple MLP es capaz de aprender a realizar la clasificación de las actividades con mucha menos cantidad de parámetros a calcular.

De esta manera, la arquitectura completa de los modelos clasificadores utilizados en este proyecto es la mostrada en la Fig. 3.3. De cada espectrograma proveniente de uno de los doce ejes de sensores se obtienen 774 características. Al concatenar las características obtenidas de los doce ejes se obtiene un vector de 9288 características, de manera que ese es el número de neuronas que presenta la capa de entrada del MLP. De este modo, cada instancia de entrada al clasificador contendrá información de los doce ejes de los SensorTag tomada en un mismo instante temporal.

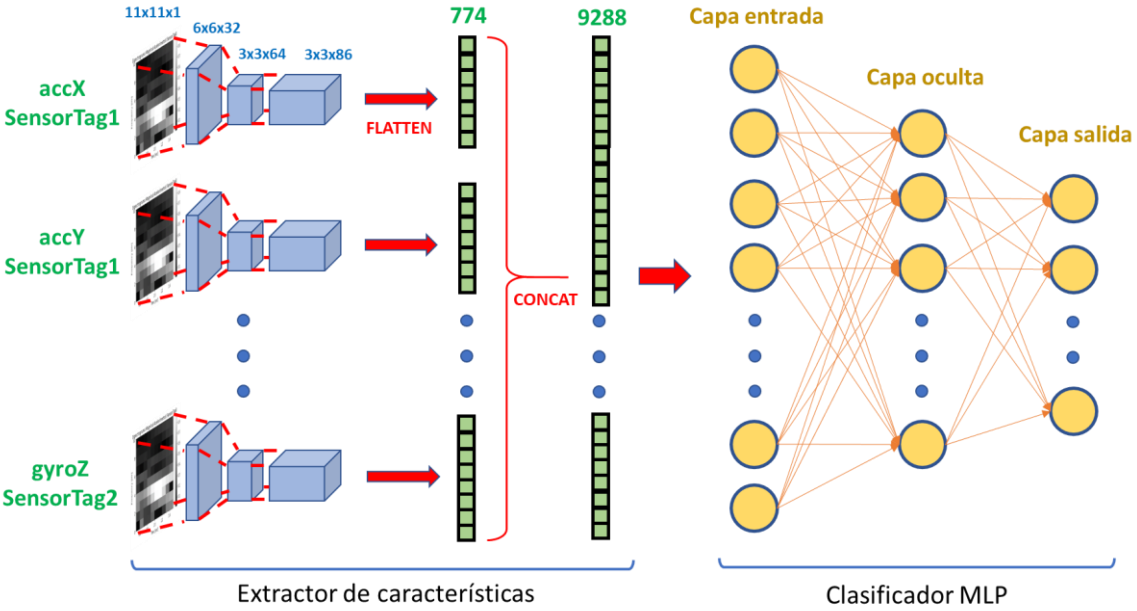



*Fig. 3.3. Arquitectura del modelo clasificador completo*

En este proyecto se han desarrollado tres modelos clasificadores diferentes que llevan a cabo funcionalidades distintas. A continuación, se resume la finalidad de cada uno de ellos, y se explicará más ampliamente en las siguientes secciones:

- **Clasificador de actividades cotidianas**: con el que se clasifican cada una de las 7 actividades cotidianas realizadas en la base de datos.
- **Filtro de golpes de tenis**: mediante el cual se clasifica si un dato de entrada es alguno de los golpes de tenis llevados a cabo en la base de datos, o si es cualquier otro movimiento posible.
- **Clasificador de golpes de tenis**: con el que se utilizan los datos filtrados como golpes de tenis por el anterior modelo para clasificarlos en alguno de los golpes de tenis recopilados en la base de datos.

## 3.3. Modelo clasificador de actividades cotidianas

En un primer paso se ha desarrollado un modelo con el que poder clasificar los datos en una de las siguientes 7 actividades cotidianas: andar, correr, saltar, agacharse/incorporarse, estar de pie, estar sentado, y sentarse/levantarse. La finalidad de este modelo es el poder comparar los resultados obtenidos con los observados en el estado del arte en la sección 1.4.5, los cuales presentaban clases similares. Las actividades a clasificar han sido seleccionadas de manera que sean algún tipo de actividad que se podría llevar a cabo en una pista de tenis, ya sea entre golpes de tenis o en un descanso. De este modo, los datos de esas actividades también son útiles a la hora de entrenar el modelo que actúa como filtro de golpes de tenis.

Tal y como se ha comentado, el MLP tendrá tantas neuronas en la capa de entrada como características obtenidas, es decir, 9288 neuronas. La capa de salida tendrá 7 neuronas ya que es el número de actividades que se desea clasificar, siendo la clase predicha aquella cuya neurona presente mayor valor a su salida. La estimación del número de neuronas ocultas es explicada en la sección 3.3.2.

### 3.3.1. Parámetros de diseño y de entrenamiento

Los pesos de las conexiones de la red son inicializados mediante la técnica de *He initialization*.

Tanto la capa de entrada, como la capa oculta de la red, están formadas por neuronas a las que se le aplica la función de activación ELU. En la Fig. 3.4 se muestra un ejemplo de estas neuronas, donde cada neurona está formada realmente por la operación de multiplicación de pesos por las entradas y la suma del umbral (bias), más la aplicación de la función de activación ELU.

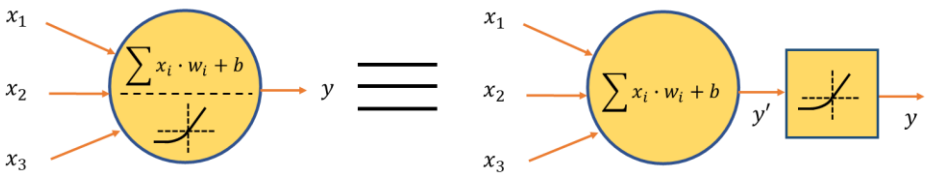



*Fig. 3.4. Neuronas de capa de entrada y capa oculta*

La capa de salida de la red está formada por neuronas a las que se le aplica la función de activación Softmax, con la cual se obtiene en cada neurona de salida la probabilidad de que el dato de entrada pertenezca a esa clase.

La función de coste utilizada es la entropía cruzada (*Cross Entropy*) ya que acelera el tiempo de convergencia al penalizar al modelo cuando obtiene una probabilidad baja para la clase objetivo. El algoritmo de entrenamiento utilizado es el descenso por el gradiente junto con la técnica de optimización *Adam*. El factor de entrenamiento utilizado es de 0.0001.

Para mejorar los resultados obtenidos, y acelerar el entrenamiento, se ha realizado entrenamiento por *mini-batch*. Antes de introducir cada lote de datos a la capa de entrada de la red, se normalizan con la media y desviación estándar de todos los datos de entrenamiento.

El número de ciclos ejecutados durante el entrenamiento es de 120, ya que es el valor a partir del cual la precisión obtenida con el conjunto de validación deja de mejorar y alcanza una zona plana, tal y como se muestra en la Fig. 3.5.

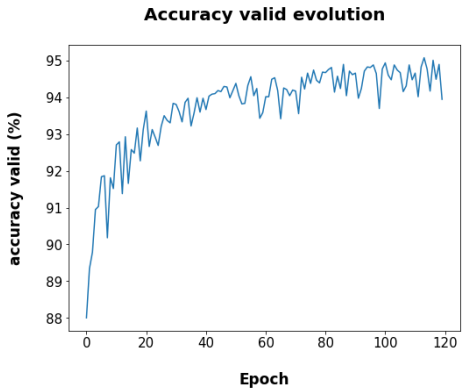



*Fig. 3.5. Ejemplo de curva de precisión durante entrenamiento de red*

### 3.3.2. Comparación de arquitecturas del clasificador

La precisión obtenida en la red depende de la arquitectura escogida para la capa oculta. Variando el número de neuronas ocultas se puede aumentar o reducir la precisión obtenida. En caso de escoger menos neuronas de las necesarias para separar las distribuciones de datos de distintas clases, las clasificaciones erróneas aumentarán. En caso de seleccionar demasiadas neuronas ocultas, aumentamos considerablemente el número de parámetros a calcular en la red, y se aumenta el riesgo de sobreajustar la red a los datos de entrenamiento.

Para la comparación de distintas arquitecturas, se ha llevado a cabo el entrenamiento de arquitecturas con distinto número de neuronas ocultas y se ha validado cada uno mediante 10 validaciones cruzadas. De este modo se consigue una mayor fiabilidad de los resultados obtenidos por cada arquitectura que si usáramos un solo conjunto de validación, ya que evitamos que la representatividad de los datos seleccionados para validación influya en la precisión obtenida.

En la Fig. 3.6 se puede observar una gráfica con las distribuciones de precisión obtenidas para cada una de las arquitecturas en las 10 validaciones cruzadas. Como se puede observar, la red que mejor resultados presenta es la de 35 neuronas ocultas ya que es la que menor dispersión de precisiones obtenidas presenta (con lo que su comportamiento es más fiable) con una precisión obtenida aceptable. Las arquitecturas con mayor número de neuronas ocultas no presentan un aumento significativo de la precisión obtenida, pero aumentan fuertemente el número de parámetros de la red, por lo que no son de interés debido a su mayor tendencia a sobreajustarse.

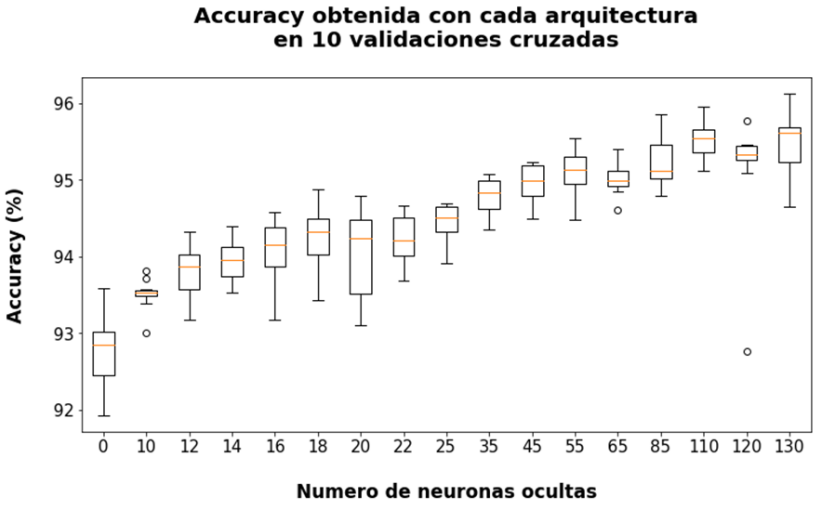



*Fig. 3.6. Comparación arquitecturas modelo clasificador de actividades cotidianas*

## 3.4. **Modelo filtrador de golpes de tenis**

Con el fin de mejorar los resultados obtenidos en el clasificador de golpes de tenis final, se ha implementado previamente un modelo que filtra los datos que son golpes de tenis de los que no. De este modo, se consigue descartar la mayoría de los datos no pertenecientes a ninguna de las clases de golpes de tenis entrenadas, introduciendo al clasificador de golpes de tenis únicamente los datos que hayan sido clasificados positivamente por este modelo.

La capa de entrada del clasificador es de 9288 neuronas. En este modelo la capa de salida está formada por una única neurona binaria, de modo que el valor de salida será cercano a uno cuando el dato de entrada pertenezca a uno de los golpes de tenis, y un cero en caso de que sea cualquier otro tipo de actividad. Para seleccionar qué datos tienen un valor suficientemente cercano a 1 como para indicarlo como golpe de tenis, se establece un valor umbral como parámetro de diseño con el cual se puede modificar los resultados de precisión y *recall* obtenidos.

En este caso no se realiza comparación de distintas arquitecturas variando el número de neuronas ocultas ya que, como se verá en la sección 4.2, las precisiones de clasificación obtenidas utilizando la arquitectura sin capa oculta son muy altas. Esto significa que añadir neuronas ocultas no mejoraría los resultados, sino que solo aumentaría el número de parámetros de la red.

### 3.4.1. **Parámetros de diseño y de entrenamiento**

Los pesos de las conexiones de la red son inicializados mediante la técnica de *He initialization.*

La función de activación de la capa de entrada es la función ELU. En el caso de la neurona de salida se utiliza la función de activación sigmoidea, de manera que la salida de la neurona nos dará un valor entre 0 y 1

correspondiente con la probabilidad de pertenencia a uno de los golpes de tenis.

La función de coste escogida es la entropía cruzada. Para el entrenamiento de la red se ha utilizado el descenso por el gradiente junto con el optimizador *Adam*, con un factor de entrenamiento de 0.0001.

Se aplica entrenamiento por *mini-batch*, y se realiza la normalización de los datos de entrada mediante el cálculo de la media y desviación estándar de todo el conjunto de entrenamiento. El número de ciclos de entrenamiento realizados es 35.

Para la selección de un valor de umbral correcto para la clasificación binaria de los golpes de tenis, se ha obtenido la curva *Precision/Recall* (PR) y la curva ROC (Fig. 3.7) con las salidas obtenidas al introducir los datos de entrenamiento. Observando la curva ROC se pone de manifiesto el buen funcionamiento del clasificador entrenado, ya que el área bajo la curva es prácticamente la unidad. Al observar la curva PR se identifica que con un valor umbral de 0.5 se consigue un valor similar en precisión y *Recall*. Sin embargo, para este modelo es preferible que todos los datos pertenecientes a golpes de tenis sean clasificados como tal, por lo que se da más importancia a obtener mayor valor de *Recall*. De este modo, se escoge un valor umbral de 0.4 con el que se obtiene un Recall del 99.2%.

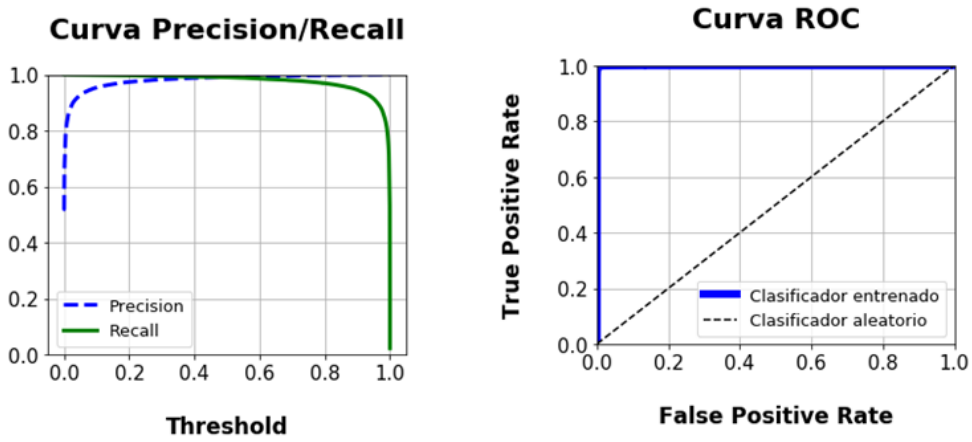



*Fig. 3.7. Curva Precision/Recall y curva ROC de filtro de golpes de tenis*

Aunque con este valor umbral la tasa de falsos positivos no sea cero, el modelo clasificador de golpes de tenis también será capaz de desecharlos tal y como se verá a continuación.

## 3.5. **Modelo clasificador de golpes de tenis**

El último modelo clasificador desarrollado con el cual se lleva a cabo el objetivo final de este proyecto es un modelo que se capaz de clasificar distintos tipos de golpes de tenis. En este caso, se han recopilado datos de 4 golpes distintos de tenis: drive, revés, mate, y globo. Sin embargo, se podrían introducir nuevos golpes en la base de datos con los que entrenar el modelo.

La clasificación final de un dato en uno de los golpes de tenis o en la clase de rechazo (no tenis) se llevará a cabo mediante el uso de este modelo junto con el modelo filtrador. En la Fig. 3.8 se puede observar el diagrama de flujo del funcionamiento del sistema completo. De esta manera, los datos de entrada a este clasificador serán aquellos que ya hayan sido filtrados como golpes de tenis.

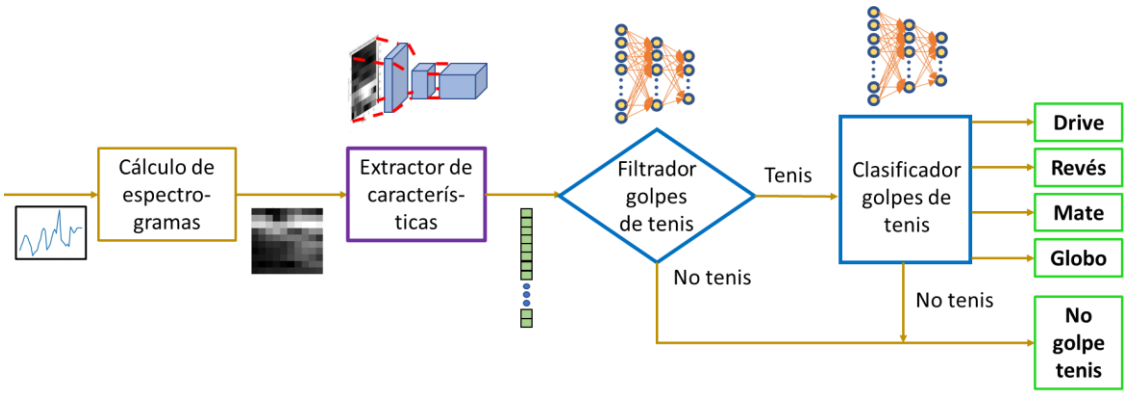



*Fig. 3.8. Diagrama de bloques del sistema clasificador de golpes de tenis completo*

Al igual que los otros modelos, la capa de entrada del MLP clasificador de golpes de tenis está formada por 9288 neuronas de entrada. La capa de salida está formada por 5 neuronas (4 para las clases de golpes de tenis, y una para la clase de rechazo). De esta manera se entrena el modelo con la base de datos completa indicando la etiqueta 1 en la clase de rechazo en caso de no ser ninguno de los golpes de tenis. De este modo, el modelo aprenderá a separar las distribuciones de datos de golpes de tenis de los datos que no lo son, de manera que cualquier otra posible actividad realizada durante un partido de tenis, y que no haya sido clasificada correctamente por el filtro, será indicada como clase de rechazo (no golpe tenis). Al igual que en el clasificador de actividades cotidianas, se realiza una comparación de distintas arquitecturas variando el número de neuronas ocultas.

### 3.5.1. Parámetros de diseño y de entrenamiento

Los pesos de las conexiones de la red son inicializados mediante la técnica de *He initialization.*

Tanto la capa de entrada, como la capa oculta de la red, están formadas por neuronas a las que se le aplica la función de activación ELU.

En las neuronas de salida se utiliza la función de activación sigmoidea, obteniendo en la salida de las neuronas un valor entre 0 y 1 correspondiente con la probabilidad de pertenencia a la clase correspondiente a cada neurona. Estas probabilidades de salida son independientes (no es como la función Softmax) de manera que te indica la probabilidad individual de pertenecer a esa clase.

De este modo como etiqueta predicha se seleccionará aquella neurona que presente mayor valor a su salida de entre las demás, siempre y cuando este valor supere un determinado valor umbral (Fig. 3.9). En caso de una clase sea seleccionada como la de mayor probabilidad, pero no supere el valor

umbral, se le asigna a la clase de rechazo, ya que es el caso en el que no te puedes fiar de las predicciones realizadas. Como valor umbral se ha seleccionado 0.05 (5%), lo que significa que nos fiamos mucho del modelo clasificador. El valor tan reducido del umbral se ha seleccionado tras comprobar que los resultados de clasificación se realizaban correctamente en la mayoría de los casos, aunque la probabilidad de salida fuera baja.

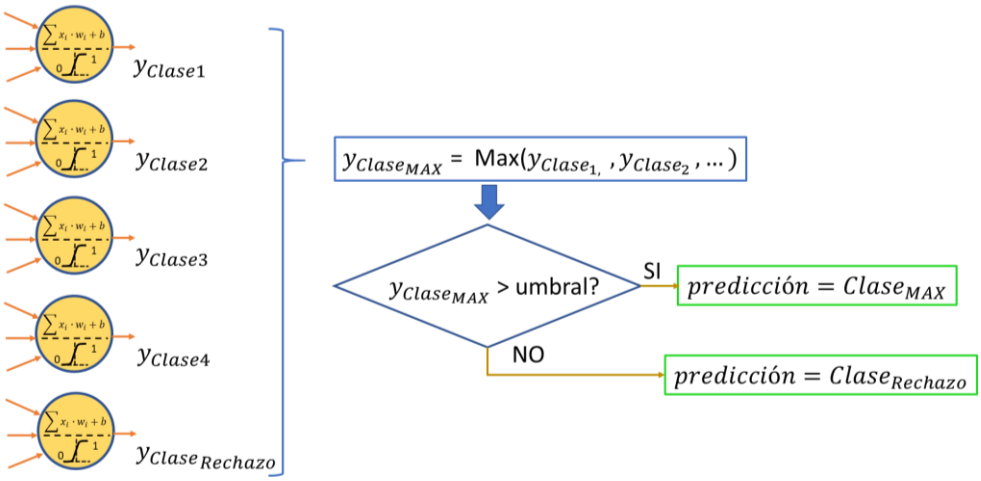



*Fig. 3.9. Selección de etiqueta predicha a partir de probabilidades de salida "y"*

La función de coste escogida es la entropía cruzada. Para el entrenamiento de la red se ha utilizado el descenso por el gradiente junto con el optimizador Adam, con un factor de entrenamiento de 0.0001.

Se aplica entrenamiento por *mini-batch*, y se realiza la normalización de los datos de entrada mediante el cálculo de la media y desviación estándar de todo el conjunto de entrenamiento. El número de ciclos de entrenamiento realizados es 45.

### 3.5.2. Comparación de arquitecturas del clasificador

Al igual que con el clasificador de actividades cotidianas, se ha llevado a cabo el entrenamiento de arquitecturas con distinto número de neuronas ocultas y se ha validado cada uno mediante 10 validaciones cruzadas.

En la Fig. 3.10 se puede observar una gráfica con las distribuciones de precisión obtenidas para cada una de las arquitecturas. En este caso, la arquitectura que mejor comportamiento muestra es la de 18 neuronas ocultas, ya que es la red con menos neuronas que presenta una baja dispersión en las validaciones, y un valor alto de precisión obtenida. La red de 22 neuronas también presenta baja dispersión, pero no mejora mucho los resultados, aumentando considerablemente el número de conexiones.

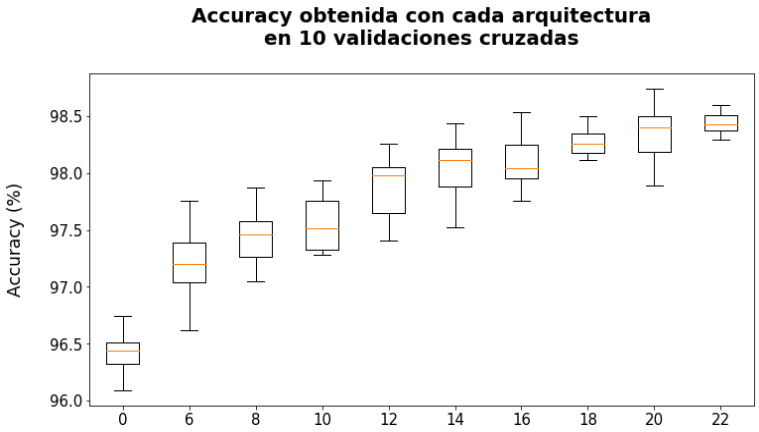



*Fig. 3.10. Comparación arquitecturas modelo clasificador de golpes de tenis*

# Capítulo 4
# Resultados de clasificación

En este capítulo se van a exponer los resultados de clasificación obtenidos con los modelos descritos en el capítulo anterior. Para la evaluación del comportamiento general que tendrán estos modelos, se llevará a cabo la clasificación de los dos conjuntos de test que se habían comentado en la sección 2.4.2.

## 4.1. Resultados del modelo clasificador de actividades cotidianas

Primero se realiza la clasificación de los datos de test tratando los espectrogramas individualmente, como si se tratara con imágenes. Con el fin de mejorar los resultados obtenidos, también se ha llevado a cabo una segunda clasificación en la que, teniendo en cuenta que las señales recopiladas son secuencias temporales, se asigna a cada dato la etiqueta más votada de entre sus 4 vecinos, tal y como se ilustra en la Fig. 4.1.

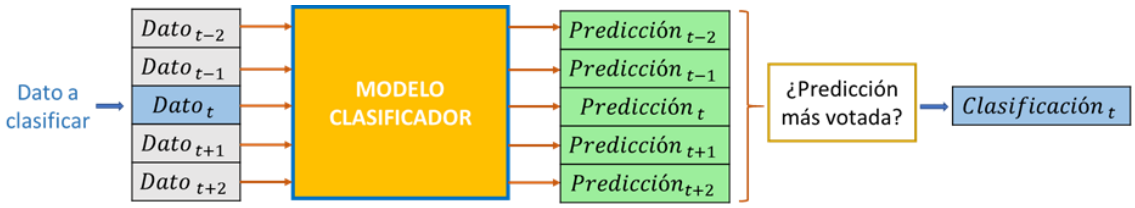



*Fig. 4.1. Clasificación por votación*

### 4.1.1. Clasificación sin votación

En la Fig. 4.2 se puede observar la matriz de confusión obtenida con el conjunto de test de los datos separados del conjunto de entrenamiento. También se muestra el valor del índice Kappa obtenido, así como las métricas de precisión y *recall* promedio de cada clase.

En la matriz de confusión se observa como la mayoría de las clasificaciones realzadas están en la diagonal principal por lo que se manifiesta el buen funcionamiento del clasificador. El índice Kappa obtenido también es indicativo del buen funcionamiento del clasificador al ser superior a 0.7, tal y como se ha explicado en la sección 1.4.5. En cada fila de la matriz

se ha representado tanto el número de datos en cada clase predicha, como el porcentaje al que corresponden del total de datos de la clase verdadera.

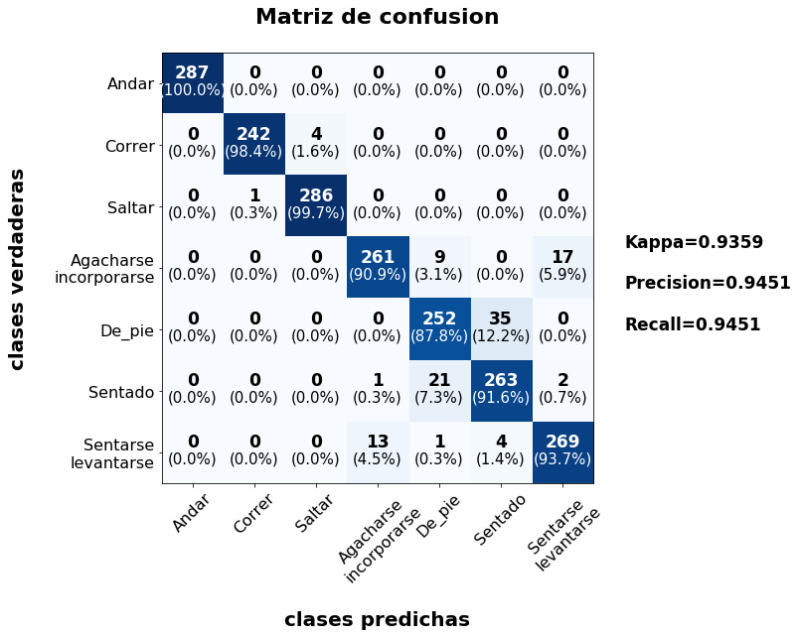



*Fig. 4.2. Matriz de confusión del clasificador de actividades cotidianas, con conjunto de test de datos separados de los de entrenamiento*

Para comprobar la capacidad de generalización que tiene el modelo, en la Fig. 4.3 se muestra las clasificaciones obtenidas de las actividades llevadas a cabo por un sujeto que no se incorporó a la base de datos de entrenamiento. Como se puede observar, en este caso el modelo no generaliza bien para un par de clases, pero observando los resultados obtenidos con los demás modelos (como se verá en las secciones siguientes), se intuye que ha sido producido por una mala ejecución de esas actividades.

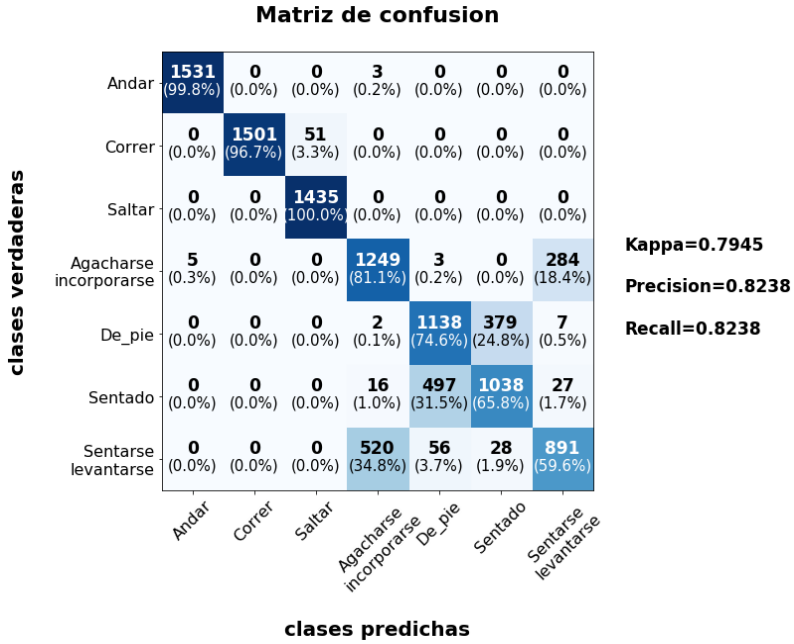



*Fig. 4.3. Matriz de confusión del clasificador de actividades cotidianas, con conjunto de test del sujeto no introducido*

Como era de esperar, las clases que más se confunden son la clase "agacharse/incorporarse" con la clase "sentarse/levantarse" debido al parecido de las actividades, y la clase "estar de pie" con la clase "estar sentado" ya que, al ser actividades estáticas, el tratamiento frecuencial de las señales con los espectrogramas no aportará diferencias entre ellas. Para que la causa de este error no sea solo una suposición, y descartar que haya sido problema del entrenamiento del clasificador, en las figuras 4.4, 4.5 y 4.6 se representa la distribución de datos de entrada al clasificador con el conjunto de entrenamiento y los dos de test utilizando el método UMAP (*Uniform Manifold Approximation and Projection*) [43].

En la Fig. 4.4, correspondiente a los datos de entrenamiento, se puede comprobar como las clases que han sido confundidas están superpuestas. De este modo, no es posible que el clasificador aprenda a separarlas completamente ya que las muestras de algunas clases están entremezcladas.

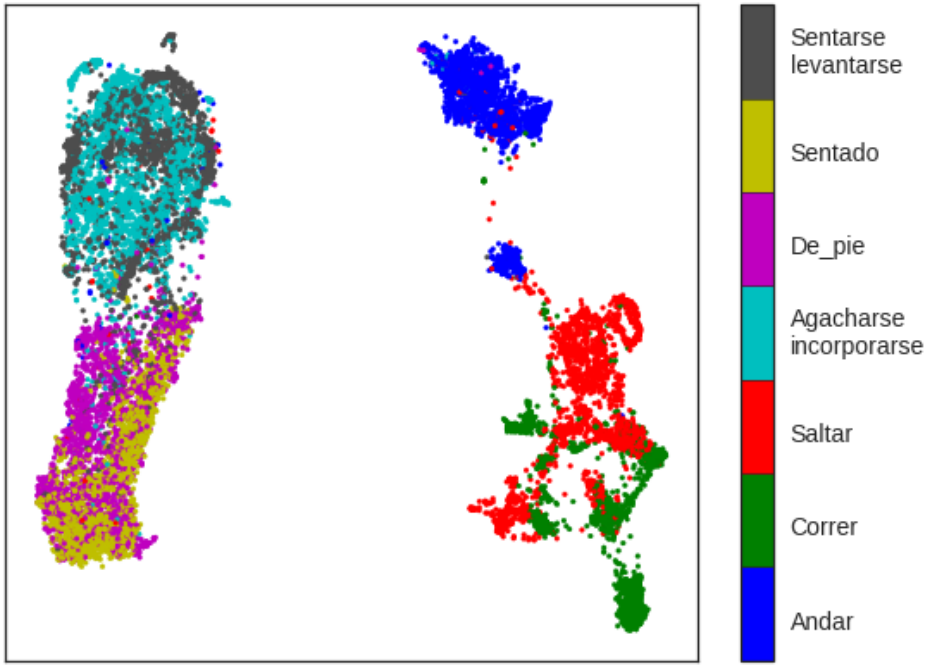



*Fig. 4.4. Distribución de características extraídas proyectadas en 2D, con el conjunto de datos de entrenamiento*

En la Fig. 4.5 se ha representado el conjunto de test de datos separados de los de entrenamiento, utilizando la misma proyección obtenida con los datos de entrenamiento. En ella se puede comprobar como la distribución de datos es parecida a la de los datos de entrenamiento, con lo que se explica los buenos resultados obtenidos en la Fig. 4.2 a pesar de las confusiones en las clases entremezcladas.

En la Fig. 4.6 se ha representado el conjunto de test del sujeto no introducido en el entrenamiento. En ella se observa cómo se cumple la suposición mencionada de que el sujeto de test ejecutó las actividades confundidas de manera irregular, ya que sus distribuciones están mucho más entremezcladas que lo casos mostrados en las figuras 4.4 y 4.5. De este modo se explica la razón del aumento de la confusión entre clases que se había visto en la matriz de confusión de la Fig. 4.3.

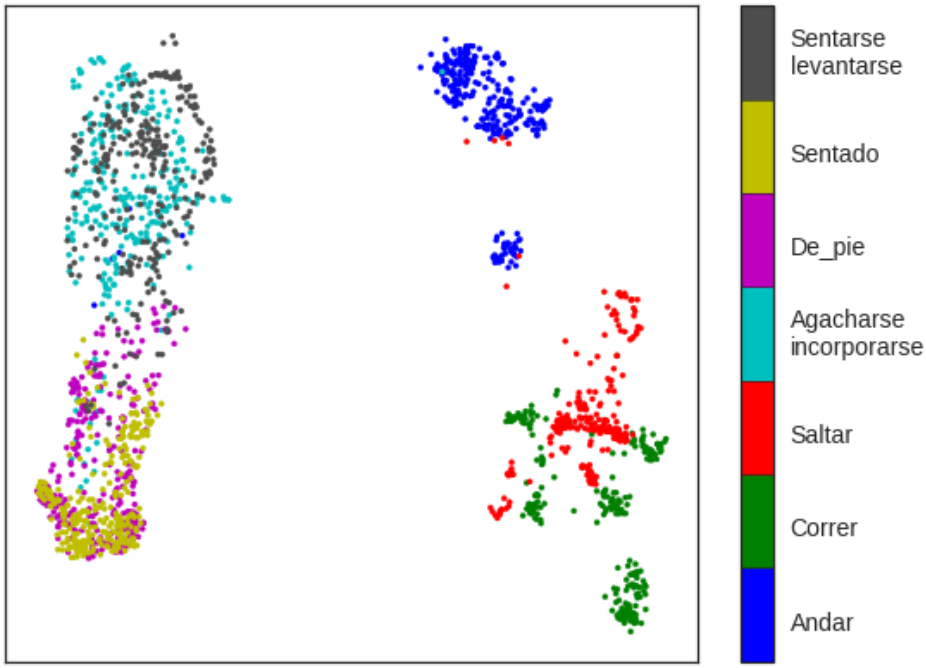



*Fig. 4.5. Distribución de características extraídas proyectadas en 2D, con el conjunto de test de datos separados de los de entrenamiento*

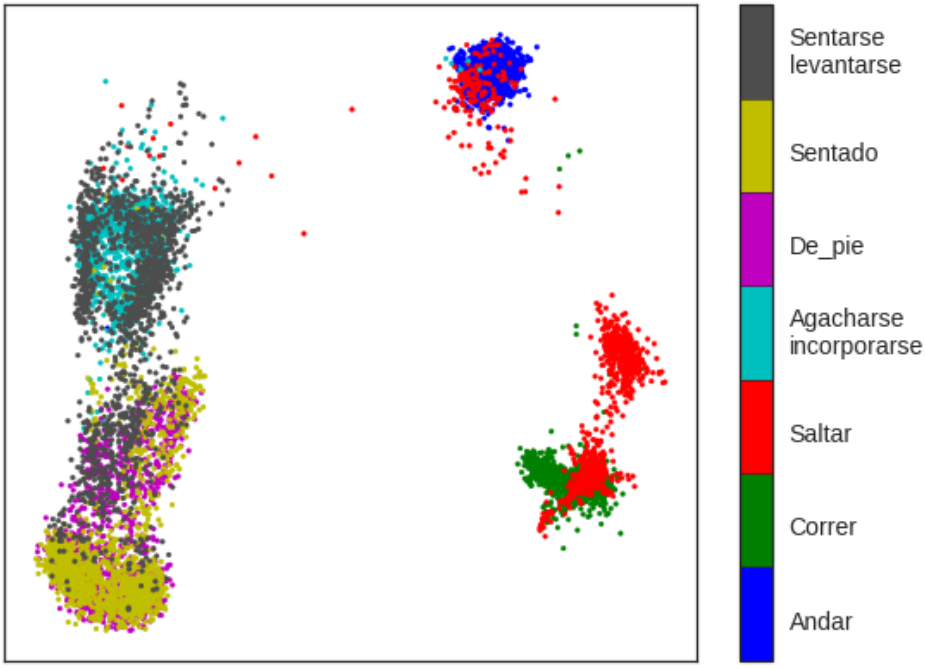



*Fig. 4.6. Distribución de características extraídas proyectadas en 2D, con el conjunto de test del sujeto no introducido durante el entrenamiento*

Adicionalmente, en el Anexo A se han expuesto las proyecciones de datos obtenidas en el caso de que se hubiesen utilizado menos sensores, destacando así la importancia que tienen ciertos sensores para llevar a cabo una mejor clasificación. Para ello se han ido eliminando conjuntos de sensores enteros, o solo algún eje de estos.

### 4.1.2. Clasificación con votación

Tal y como se ha comentado al inicio de este capítulo, también se ha llevado a cabo la clasificación de las actividades teniendo en cuenta las predicciones realizadas sobre los datos contiguos en la secuencia temporal, con el fin de obtener mejores resultados en la clasificación.

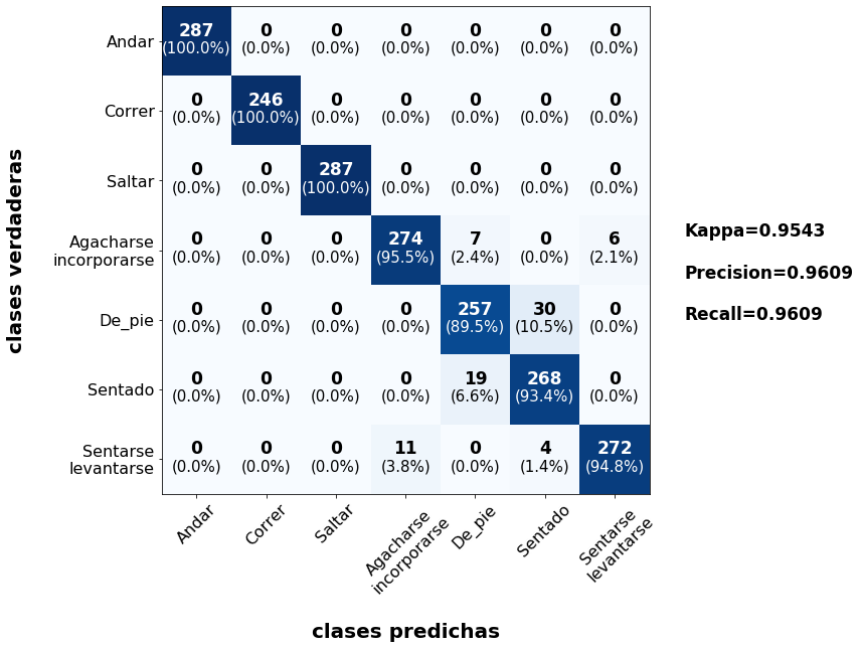



*Fig. 4.7. Matriz de confusión del clasificador de actividades cotidianas, con conjunto de test de datos separados de los de entrenamiento, y votación*

En las figuras 4.7 y 4.8 se muestran las nuevas matrices de confusión obtenidas, donde se puede apreciar la mejora de los resultados respecto a la clasificación sin votación mostrada en las figuras 4.2 y 4.3.

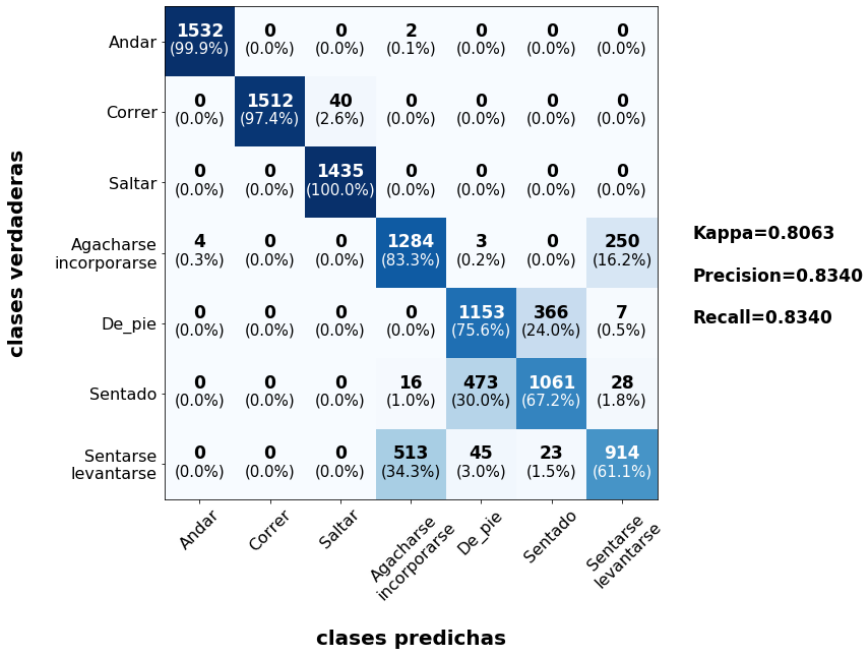
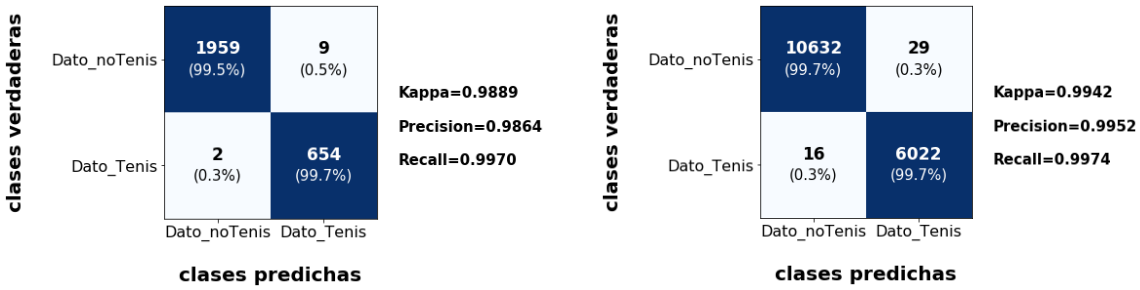



*Fig. 4.8. Matriz de confusión del clasificador de actividades cotidianas, con conjunto de test del sujeto no introducido, y votación*

## 4.2. Resultados del modelo filtrador de golpes de tenis

Tras comprobar que la clasificación realizada con votación de entre 4 vecinos mejora los resultados obtenidos, en los siguientes modelos ya se van a exponer los resultados obtenidos al aplicar el proceso de votación.

En la figura 4.9 se muestran los resultados obtenidos en la clasificación por el modelo filtrador de golpes de tenis con los dos conjuntos de test.




*a)* *b)*

*Fig. 4.9. Matriz de confusión del modelo filtrador de golpes de tenis utilizando votación. a) Conjunto de test de datos separados de los de entrenamiento. b) Conjunto de test del sujeto no introducido.*

Como se puede observar, debido al valor umbral escogido el valor de *recall* que se obtiene es muy alto, ya que casi todos los datos que son golpes de tenis son clasificados como tal. Aunque se clasifiquen varios datos que no son golpes de tenis como que sí lo son, el modelo clasificador de golpes de tenis también puede desecharlos por lo que es preferible obtener un *recall* alto.

En la Fig. 4.9 b) se comprueba la buena generalización realizada por el modelo, ya que la clasificación realizada de los datos de un sujeto no visto durante el entrenamiento presenta mejores resultados incluso que el conjunto de test separado de los datos de entrenamiento.

Una vez clasificados los datos, únicamente se introducirán al modelo clasificador de golpes de tenis los que se hayan predicho como "*Dato_Tenis*". De este modo se introducirán 11 datos erróneos y 653 correctos en el caso del conjunto de test de datos separados de los de entrenamiento, y 40 datos erróneos y 6019 correctos en el caso del conjunto de test del usuario no introducido en el entrenamiento.

## 4.3. Resultados del modelo clasificador de golpes de tenis

En las figuras 4.10 y 4.11 se puede observar los resultados obtenidos por el clasificador de golpes de tenis. En ellas únicamente se muestran los datos que han pasado el filtro aplicado con el anterior modelo. En el caso de la base de datos de test separada de los datos de entrenamiento (Fig. 4.10), se observa que la clasificación es prácticamente correcta en su totalidad.

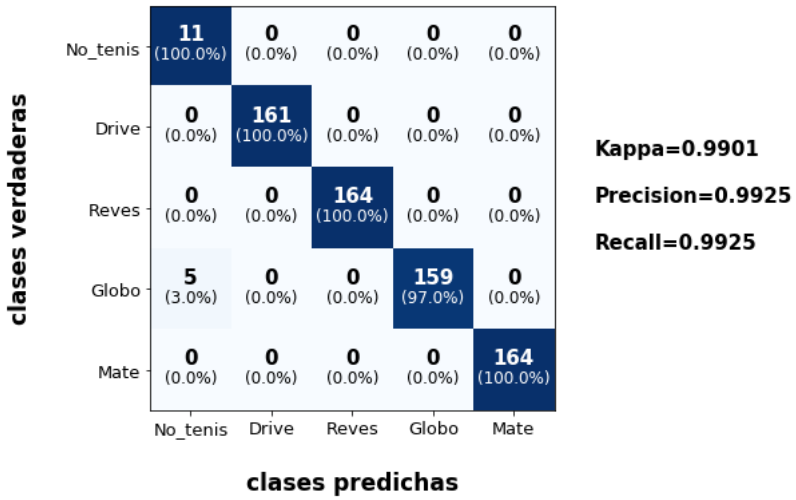



*Fig. 4.10. Matriz de confusión del modelo clasificador de golpes de tenis, con conjunto de test de datos separados de los de entrenamiento, y votación*

En la Fig. 4.11 correspondiente con los datos del sujeto no introducido a la red, se ve como los pocos datos no pertenecientes a ninguno de los golpes de tenis (que habían sido clasificados incorrectamente por el modelo filtrador) se clasifican correctamente. El resto de las clases se clasifican con niveles de precisión por encima del 97%, exceptuando la clase globo que presenta una precisión del 91,6%. La confusión de esta clase puede deberse a que dicho usuario no realizaba los golpes de manera correcta, confundiéndose así con otros movimientos. A pesar de ello, al obtener un índice Kappa del 95.55% se demuestra la robustez del modelo obtenido, superando así el objetivo planteado para el proyecto.

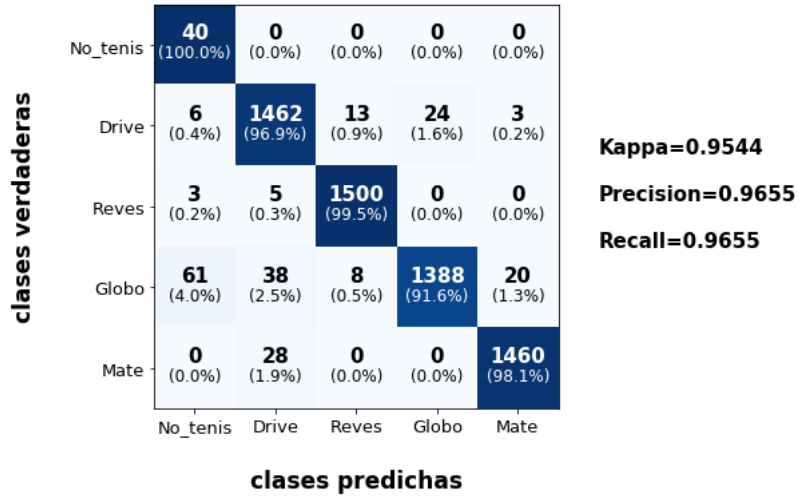



*Fig. 4.11. Matriz de confusión del modelo clasificador de golpes de tenis, con conjunto de test del sujeto no introducido, y votación*

# Capítulo 5
# Conclusiones y líneas futuras

En este capítulo se van a exponer las conclusiones obtenidas tras el desarrollo del proyecto. Se van a destacar los pasos seguidos para alcanzar los resultados obtenidos, y se comparan éstos con los objetivos que se habían marcado. Finalmente se expondrán las posibles líneas futuras de continuación al proyecto realizado.

## 5.1. Conclusiones

En este proyecto se ha llevado a cabo el desarrollo de un sistema clasificador de golpes tenis el cual presente robustez ante las dimensiones corporales, peso, y sexo de los sujetos.

Para ello, se ha comenzado realizando un estudio del estado del arte para comprobar que técnicas se usan habitualmente para sistemas de reconocimiento de actividades humanas, y cuáles de ellas presentan invariancia ante las posibles características de cualquier sujeto.

Tras comprobar que no se dispone de una base de datos pública con la que poder llevar a cabo los objetivos deseados, se ha optado por el desarrollo de una base de datos propia. De este modo se abre la posibilidad de inclusión de nuevos movimientos de tenis con los que entrenar el modelo en una futura ampliación al proyecto. Para realizar la base de datos se ha llevado a cabo la selección y configuración de dos dispositivos portables de bajo consumo, de los que se ha obtenido información de su acelerómetro y giróscopo.

Para tratar de eliminar la dependencia del rendimiento del modelo con las características físicas de los sujetos que realizaron la base de datos, se ha llevado a cabo el procesamiento de las secuencias temporales de datos de los sensores mediante el cálculo de espectrogramas, los cuales también proporcionan información de la evolución frecuencial en el tiempo.

Para la labor de extracción de características y clasificación de los datos, se han desarrollado modelos neuronales con aprendizaje semi-supervisado a partir de las librerías de TensorFlow. Como extractor de características se ha entrenado una red “Autoencoder Convolucional” de manera no supervisada, consiguiendo reducir enormemente la cantidad de datos etiquetados necesarios a la hora de entrenar un clasificador. La red de clasificación seleccionada ha sido el Perceptrón Multicapa (MLP). De este modo, se han desarrollado tres modelos clasificadores distintos modificando la arquitectura del MLP. El primero de ellos se encarga de clasificar 7 actividades cotidianas

que también pueden ser realizadas durante un partido de tenis, y ha sido utilizado para comparar los resultados obtenidos con los vistos en el estado del arte. El segundo y tercer modelo se usan conjuntamente para obtener los mejores resultados posibles en la detección de uno de los 4 golpes de tenis entrenados. El segundo clasificador actúa como filtro para saber si un movimiento pertenece a una de las clases de golpes, mientras que el tercero lleva a cabo la clasificación de los datos filtrados en uno de los distintos golpes de tenis.

Se ha comprobado como los resultados obtenidos con el clasificador de actividades cotidianas, en el caso del conjunto de test de datos separados de los de entrenamiento, están en niveles parecidos a los vistos en el estado del arte (Tabla III), obteniendo una precisión del 96,09%. Este tipo de conjunto de test es el utilizado habitualmente en las investigaciones encontradas. En el caso de los datos procedentes de una persona que no ha sido vista nunca en el entrenamiento del modelo, se ha obtenido una precisión del 82,38%, que es algo más reducido debido a las complicaciones comentadas, pero es lo suficientemente preciso como para presentar cierta robustez y cumplir con el objetivo propuesto.

En el sistema clasificador de golpes de tenis se ha obtenido una precisión total del 99,25% con la base de datos de test con personas que ya habían sido vistas por el modelo, y del 96,55% para el sujeto nuevo, con lo que se manifiesta el buen funcionamiento y robustez del sistema clasificador desarrollado, el cual era el objetivo principal de este proyecto.

## 5.2. **Líneas futuras**

Como clara línea futura de ampliación a este proyecto, sería la captación de datos de manera *on-line* con la cual poder realizar la recopilación de estadísticas de golpes en tiempo real.

También se plantea la posible ampliación de la base de datos para que el sistema clasificador sea capaz reconocer la mayoría de los movimientos realizados durante el desempeño de un partido de tenis. Sería conveniente, para una mejora de los resultados obtenidos, que la recopilación de datos fuera realizada por personas con amplios conocimientos sobre la técnica correcta de realizar los golpes de tenis, ya que en este proyecto los sujetos carecían de ella, lo que ha podido conducir a errores en la clasificación.

Otra posible línea futura a largo plazo sería la sincronización de los datos obtenidos por los sensores con un sistema de captación de video colocado en el entorno capaz de hacer un seguimiento del esqueleto del jugador. De este modo, no sólo se podrían obtener estadísticas de golpes realizados, sino también añadir otro modelo con el que obtener estadísticas de la técnica de ejecución del movimiento realizado. Por lo tanto, el jugador podría obtener una retroalimentación de cada clase de golpe que realiza con la que mejorar su nivel de juego.

# Capítulo 6
# Referencias


[1] Z. Wang, Z. Yang, and T. Dong, "A review of wearable technologies for elderly care that can accurately track indoor position, recognize physical activities and monitor vital signs in real time," *Sensors (Switzerland)*, vol. 17, no. 2, 2017.

[2] S. A. Moore *et al.*, "Wristband Accelerometers to motiVate arm Exercise after Stroke (WAVES): Study protocol for a pilot randomized controlled trial," *Trials*, vol. 17, no. 1, pp. 1–9, 2016.

[3] S. Patel, H. Park, P. Bonato, L. Chan, and M. Rodgers, "A review of wearable sensors and systems with application in rehabilitation," *J. Neuroeng. Rehabil.*, vol. 9, no. 1, p. 21, 2012.

[4] S. C. Mukhopadhyay, "Wearable Sensors for Human Activity Monitoring: A Review," *IEEE Sens. J.*, vol. 15, no. 3, pp. 1321–1330, 2015.

[5] T. T. Um, V. Babakeshizadeh, and D. Kulić, "Exercise Motion Classification from Large-Scale Wearable Sensor Data Using Convolutional Neural Networks," *Int. Conf. on Intelligent Robots and Systems*, 2017.

[6] H. Ghasemzadeh, V. Loseu, E. Guenterberg, and R. Jafari, "Sport training using body sensor networks: a statistical approach to measure wrist rotation for golf swing," *Proc. 4th Int. ICST Conf. Body Area Networks*, 2009.

[7] L. Buthe, U. Blanke, H. Capkevics, and G. Troster, "A wearable sensing system for timing analysis in tennis," *BSN 2016 - 13th Annu. Body Sens. Networks Conf.*, pp. 43–48, 2016.

[8] D. Lara and M. a Labrador, "A Survey on Human Activity Recognition using Wearable Sensors," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 3, pp. 1192–1209, 2013.

[9] A. Bulling, U. Blanke, and B. Schiele, "A tutorial on human activity recognition using body-worn inertial sensors," *ACM Comput. Surv.*, vol. 46, no. 3, pp. 1–33, 2014.

[10] S.-R. Ke, H. Thuc, Y.-J. Lee, J.-N. Hwang, J.-H. Yoo, and K.-H. Choi, "A Review on Video-Based Human Activity Recognition, ", *Computers*, vol. 2, no. 2. 2013.

[11] P. Kumari, L. Mathew, and P. Syal, "Increasing trend of wearables and multimodal interface for human activity monitoring: A review," *Biosens. Bioelectron.*, vol. 90, no. September 2016, pp. 298–307, 2017.

[12] H. G. Espinosa, J. Lee, and D. A. James, "The Inertial Sensor: a Base Platform for Wider Adoption in Sports Science Applications," *J. Fit. Res.*, vol. 4, no. 1, pp. 13–20, 2015.

[13] S. Elies, "Performance Analysis of Commercial Accelerometers : A Parameter Review," *Sensors & Transducers*, vol. 193, no. 10, pp. 179–190, 2015.

[14] C. C. Yang and Y. L. Hsu, "A review of accelerometry-based wearable motion detectors for physical activity monitoring," *Sensors*, vol. 10, no. 8, pp. 7772–

7788, 2010.

[15] M. Shoaib, S. Bosch, O. Durmaz Incel, H. Scholten, and P. J. M. Havinga, "Fusion of smartphone motion sensors for physical activity recognition," *Sensors*, vol. 14, no. 6. 2014.

[16] H. F. Nweke, Y. W. Teh, M. A. Al-garadi, and U. R. Alo, "Deep learning algorithms for human activity recognition using mobile and wearable sensor networks: State of the art and research challenges," *Expert Syst. Appl.*, vol. 105, pp. 233–261, 2018.

[17] D. Ravi, C. Wong, B. Lo, and G.-Z. Yang, "Deep learning for human activity recognition: A resource efficient implementation on low-power devices," *2016 IEEE 13th Int. Conf. Wearable Implant. Body Sens. Networks*, pp. 71–76, 2016.

[18] J. Wang, Y. Chen, S. Hao, X. Peng, and L. Hu, "Deep learning for sensor-based activity recognition: A Survey," *Pattern Recognit. Lett.*, vol. 0, pp. 1–9, 2018.

[19] M. G. Tsipouras, A. T. Tzallas, G. Rigas, S. Tsouli, D. I. Fotiadis, and S. Konitsiotis, "An automated methodology for levodopa-induced dyskinesia: Assessment based on gyroscope and accelerometer signals," *Artif. Intell. Med.*, vol. 55, no. 2, pp. 127–135, 2012.

[20] J. P. Bello, L. Daudet, S. A. andC. Duxbury, M. Davies, and M. Sandler, "A Tutorial on Onset Detection in Musical Signals," *{IEEE} Trans. Speech Audio Process.*, vol. 13, no. 5, pp. 1035–1047, 2005.

[21] E. C. Orosco and N. M. Lopez, "Biomedical Signal Processing and Control Bispectrum-based features classification for myoelectric control," *Biomed. Signal Process. Control*, vol. 8, no. 2, pp. 153–168, 2013.

[22] O. Banos, J.-M. Galvez, M. Damas, H. Pomares, and I. Rojas, "Window Size Impact in Human Activity Recognition," *Sensors*, vol. 14, no. 4, pp. 6474–6499, 2014.

[23] M. H. M. Noor, Z. Salcic, and K. I. K. Wang, "Adaptive sliding window segmentation for physical activity recognition using a single tri-axial accelerometer," *Pervasive Mob. Comput.*, vol. 38, pp. 41–59, 2017.

[24] F. Attal, S. Mohammed, M. Dedabrishvili, F. Chamroukhi, L. Oukhellou, and Y. Amirat, "Physical Human Activity Recognition Using Wearable Sensors," *Sensors*, vol. 15, no. 12, pp. 31314–31338, 2015.

[25] D. Figo, P. C. Diniz, D. R. Ferreira, and J. M. P. Cardoso, "Preprocessing techniques for context recognition from accelerometer data," *Pers. Ubiquitous Comput.*, vol. 14, no. 7, pp. 645–662, 2010.

[26] A. Subasi and M. I. Gursoy, "EEG signal classification using PCA, ICA, LDA and support vector machines," *Expert Syst. Appl.*, vol. 37, no. 12, pp. 8659–8666, 2010.

[27] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and F. F. Li, "Large-scale video classification with convolutional neural networks," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 1725–1732, 2014.

[28] Y. Lecun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[29] F. J. Ordóñez and D. Roggen, "Deep convolutional and LSTM recurrent neural networks for multimodal wearable activity recognition," *Sensors (Switzerland)*, vol. 16, no. 1, 2016.

[30] D. Ravi, C. Wong, B. Lo, and G.-Z. Yang, "A Deep Learning Approach to on-Node Sensor Data Analytics for Mobile or Wearable Devices," *IEEE J. Biomed. Heal. Informatics*, vol. 21, no. 1, pp. 56–64, 2017.

[31] C. A. Ronao and S. B. Cho, "Human activity recognition with smartphone sensors using deep learning neural networks," *Expert Syst. Appl.*, vol. 59, pp.

235–244, 2016.
[32] W. Jiang and Z. Yin, “Human Activity Recognition Using Wearable Sensors by Deep Convolutional Neural Networks,” *Proc. 23rd ACM Int. Conf. Multimed. - MM ’15*, pp. 1307–1310, 2015.
[33] J. B. Yang, M. N. Nguyen, P. P. San, X. L. Li, and S. Krishnaswamy, “Deep Convolutional Neural Networks On Multichannel Time Series For Human Activity Recognition,” *Int. Joint Conf. on Artificial Intelligence*, pp. 3995–4001, 2014.
[34] O. Abdel-Hamid, A. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu, “Convolutional Neural Networks for Speech Recognition,” *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 22, no. 10, pp. 1533–1545, 2014.
[35] M. M. Al, R. Yakoub, B. Mansour, A. Zuair, E. Othman, and B. Benjdira, “Convolutional Neural Networks for Electrocardiogram Classification,” *J. Med. Biol. Eng.*, no. 0123456789, 2018.
[36] B. Martín-del-Brío, “Diapositivas Tema 2 de asignatura Redes Neuronales Artificiles del máster en ingeniería electrónica.,” *Univ. Zaragoza*, 2018.
[37] Texas Instrument, “CC26x0 SimpleLink ™ Developer ’ s Guide,” 2018. [Online]. Available: http://www.ti.com/lit/ug/swru393e/swru393e.pdf.
[38] Texas Instrument, “CC2650 SensorTag User’s Guide,” 2018. [Online]. Available: http://processors.wiki.ti.com/index.php/CC2650_SensorTag_User’s_Guide#Connecting_to_multiple_SensorTags.
[39] Texas Instrument, “Debug DevPack User Guide.” [Online]. Available: http://processors.wiki.ti.com/index.php/Debug_DevPack_User_Guide.
[40] InvenSense, “MPU9250 Datasheet,” 2016. [Online]. Available: https://store.invensense.com/Datasheets/invensense/RM-MPU-9250A-00.pdf.
[41] A. Géron, “Hands-On Machine Learning with Scikit-Learn and TensorFlow,” *O’Reilly*, 2017.
[42] M. Kumar Mandal, “Implementing PCA, Feedforward and Convolutional Autoencoders and using it for Image Reconstruction, Retrieval & Compression,” *Manash’s blog*. [Online]. Available: https://blog.manash.me/implementing-pca-feedforward-and-convolutional-autoencoders-and-using-it-for-image-reconstruction-8ee44198ea55.
[43] McInnes, L, Healy, J, “UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction,” *ArXiv e-prints 1802.03426*, 2018. [Online]. Code available: https://github.com/lmcinnes/umap

# Anexo A
# Proyecciones 2D UMAP de características de los datos al utilizar menos sensores

En este anexo se va a representar las proyecciones UMAP obtenidas de las características extraídas de los datos en el caso de reducir el número de sensores utilizados. Con ello, se consigue comprobar cómo de relevante es la información proporcionada por ciertos sensores para llevar a cabo una mejor clasificación. De este modo, se han realizado tres pruebas distintas en las que se han utilizado menos de 12 ejes de sensores.

## A.1. **Prueba 1**

En la primera prueba, se ha eliminado la información proporcionada por el SensorTag colocado en la cintura (sus 3 ejes de acelerómetro y sus 3 ejes de giróscopo) por lo que únicamente se dispone de la información proporcionada por el SensorTag colocado en la muñeca de los sujetos. De esta manera, se ha pasado de tener 9288 características (por cada instancia de entrenamiento) a 4644 características.

En la Fig. A.1 se muestra la proyección obtenida con los datos de entrenamiento. En ella observamos como las distintas clases presentan distribuciones mucho menos separadas unas de otras que en el caso de utilizar los 12 ejes (Fig. 4.4). Sin embargo, se siguen intuyendo regiones de decisión claras entre la mayoría de las clases. Comparando esta figura con la Fig. 4.4 se observa como el SensorTag colocado en la cintura tiene especial relevancia para separar las clases "andar", "correr" y "saltar" del resto, aunque también provoca que correr y saltar se confundan más.

En las figuras A.2 y A.3 se han representado los dos conjuntos de test con la misma proyección obtenida con los datos de entrenamiento. Como se puede observar en las tres figuras mostradas, al quitar el SensorTag de la cintura también sigue estando presente la confusión entre las clases "Agacharse/Incorporarse" con "Sentarse/Levantarse", y las clases "De pie" y "Sentado". En este caso también se ve en las distribuciones de los conjuntos de test, como las clases andar y saltar tienen regiones muy próximas a las clases Sentarse/Levantarse y "Agacharse/Incorporarse".

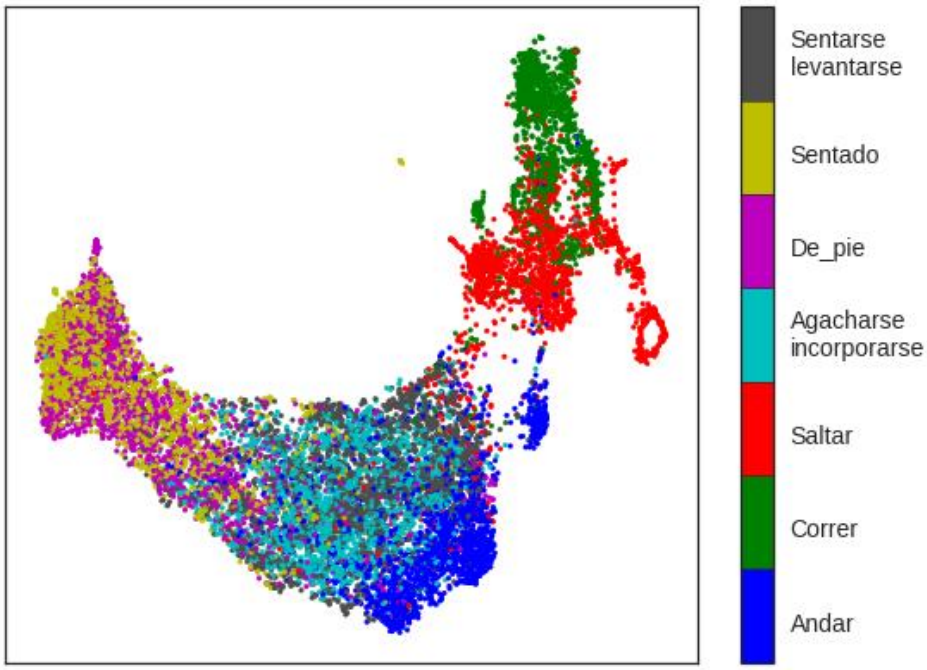



*Fig. A.1. Distribución de características extraídas proyectadas en 2D, con el conjunto de entrenamiento, en la Prueba 1*

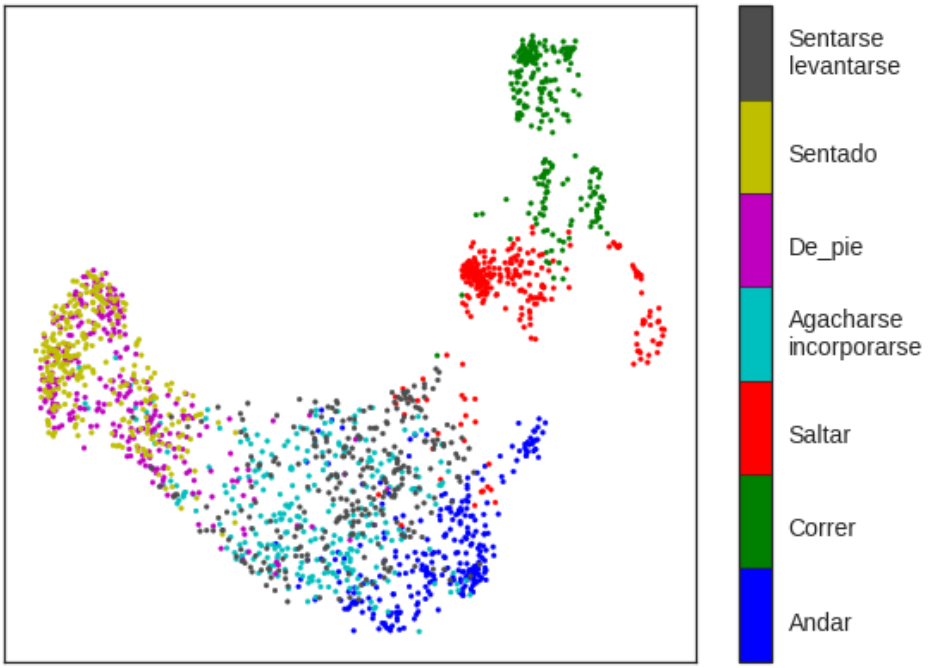



*Fig. A.2. Distribución de características extraídas proyectadas en 2D, con el conjunto de test separado de los datos de entrenamiento, en la Prueba 1*

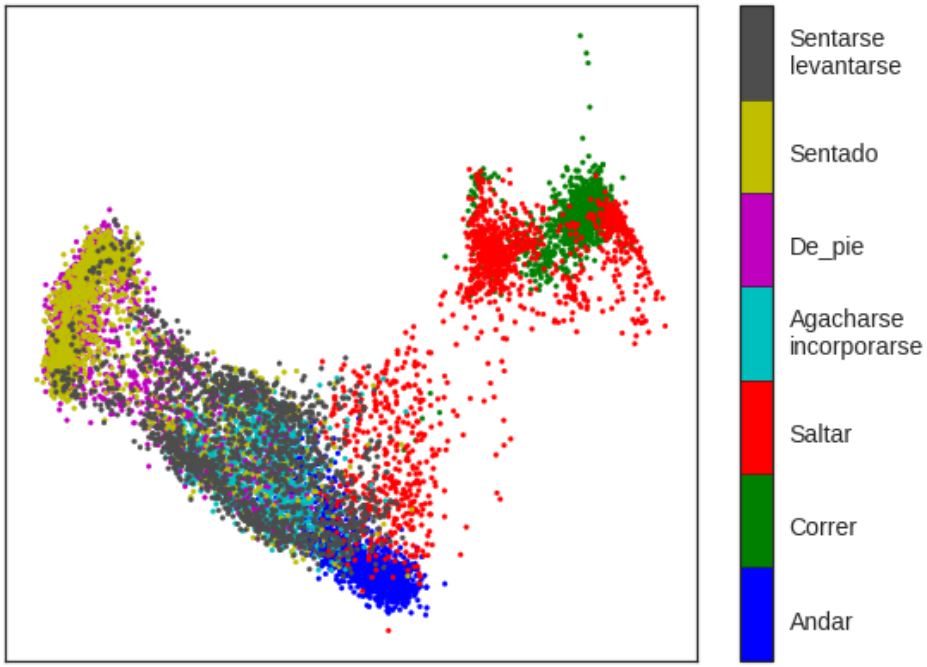



*Fig. A.3. Distribución de características extraídas proyectadas en 2D, con el conjunto de test de sujeto no introducido durante el entrenamiento, en la Prueba 1*

## A.2. **Prueba 2**

En la segunda prueba se ha realizado el caso inverso al anterior, eliminando la información proporcionada por el SensorTag colocado en la muñeca (sus 3 ejes de acelerómetro y sus 3 ejes de giróscopo), por lo que únicamente se dispone de la información proporcionada por el SensorTag colocado en la cintura de los sujetos. De esta manera, se ha pasado de tener 9288 características (por cada instancia de entrenamiento) a 4644 características.

Al proyectar el espacio de 4644 dimensiones sobre 2D con UMAP (figuras A.4, A.5, y A.6), se confirma lo que se había descrito en la Prueba 1 al comparar sus resultados con los de la Fig. 4.4. La información que proporciona el SensorTag de la cintura con todos sus ejes es importante para diferenciar las clases andar, correr y saltar del resto, pero provoca que estas dos últimas se entremezclen más. Las clases que más confusión creaban entre ellas siguen sin ser más separables.

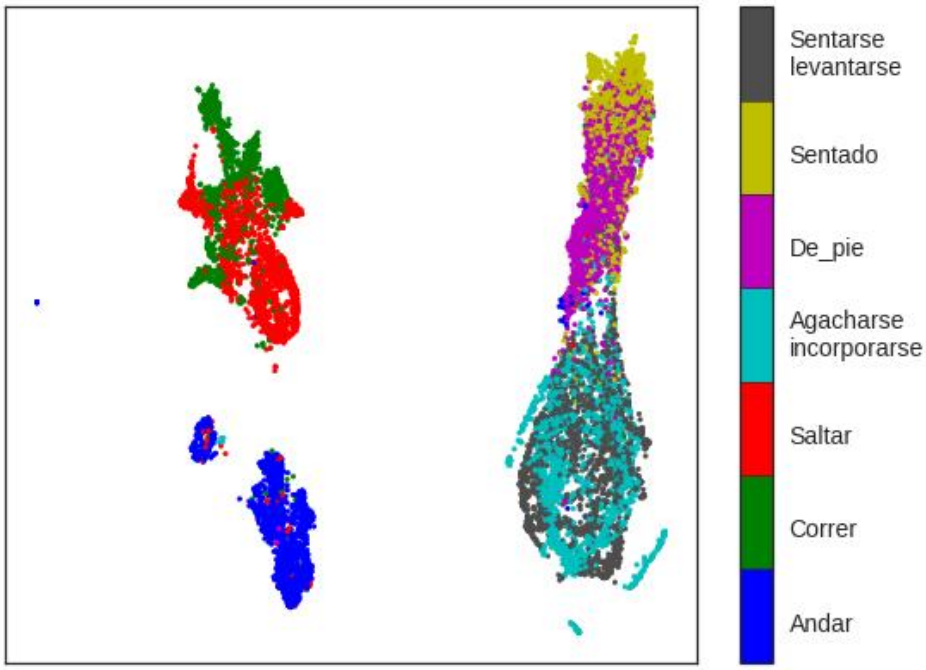



*Fig. A.4. Distribución de características extraídas proyectadas en 2D, con el conjunto de entrenamiento, en la Prueba 2*

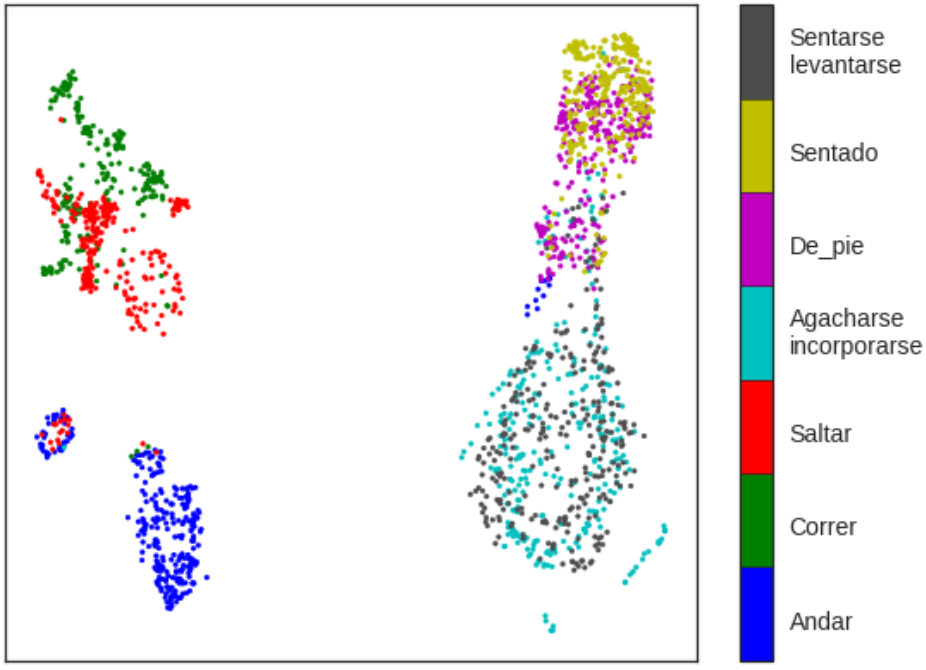



*Fig. A.5. Distribución de características extraídas proyectadas en 2D, con el conjunto de test separado de los datos de entrenamiento, en la Prueba 2*

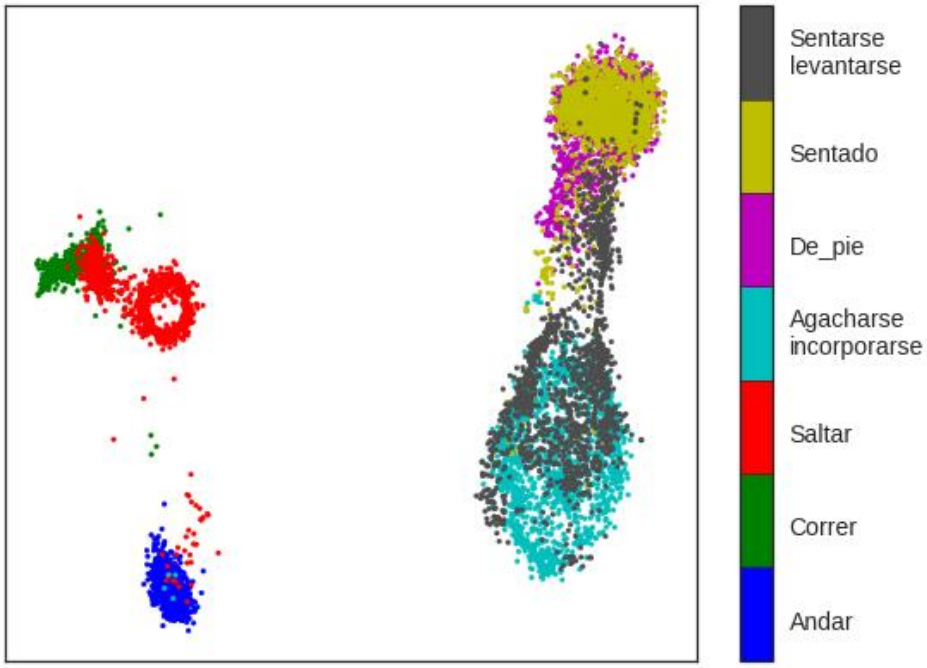



*Fig. A.6. Distribución de características extraídas proyectadas en 2D, con el conjunto de test de sujeto no introducido durante el entrenamiento, en la Prueba 2*

## A.3. **Prueba 3**

En la tercera prueba se utilizan únicamente los acelerómetros de ambos SensorTag. De esta manera, se ha pasado de tener 9288 características (por cada instancia de entrenamiento) a 4644 características.

Al comparar la proyección del espacio de 4644 dimensiones sobre 2D con (figuras A.7, A.8, y A.9) con la Fig 4.4 donde se disponía de todos los sensores, se comprueba que la información más relevante es la proporcionada por los acelerómetros y no por los giróscopos, ya que las distribuciones son mucho más parecidas en este caso que en las otras dos pruebas anteriores. También se aprecia cómo sin giróscopos las clases con más confusión se han separado un poco más.

Al igual que ocurría en la Fig. 4.6, con el conjunto de datos de test del usuario no introducido durante el entrenamiento, la clase "Agacharse/Incorporarse" está más dispersa que en el caso del otro conjunto de test y el conjunto de entrenamiento por lo que obtendrá peores resultados.

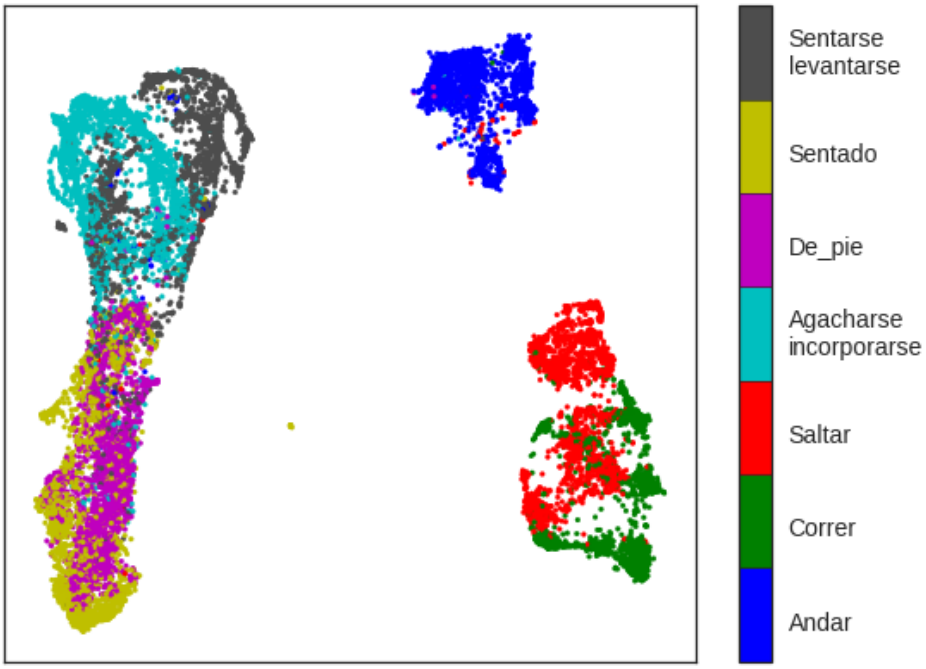



*Fig. A.7. Distribución de características extraídas proyectadas en 2D, con el conjunto de entrenamiento, en la Prueba 3*

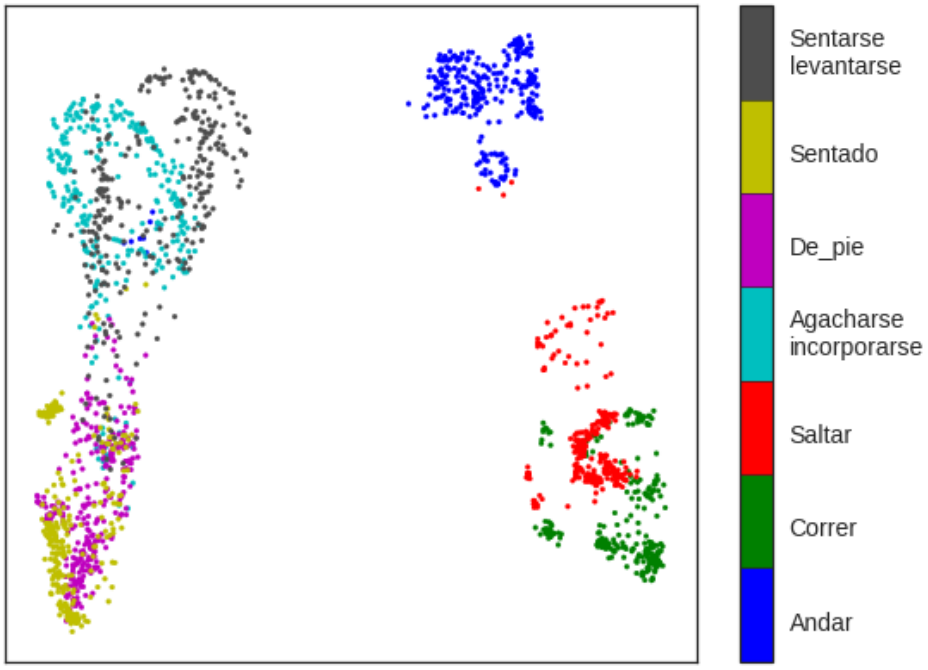



*Fig. A.8. Distribución de características extraídas proyectadas en 2D, con el conjunto de test separado de los datos de entrenamiento, en la Prueba 3*

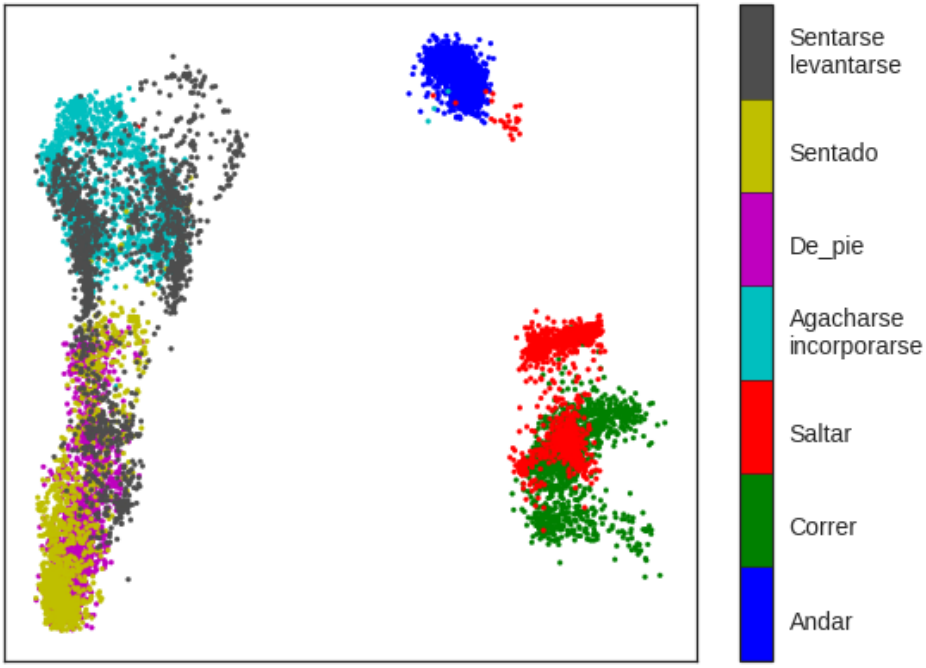



*Fig. A.9. Distribución de características extraídas proyectadas en 2D, con el conjunto de test de sujeto no introducido durante el entrenamiento, en la Prueba 3*