RECONOCIMIENTO DE ACCIONES DE MOVIMIENTO HUMANAS CON UN SMARTPHONE MEDIANTE APRENDIZAJE PROFUNDO

RECOGNITION OF HUMAN MOVEMENT ACTIONS WITH A SMARTPHONE USING DEEP LEARNING

TRABAJO DE FIN DE GRADO
CURSO 2021-2022

AUTOR
EMMA TALAVERA RODRÍGUEZ

DIRECTOR
GONZALO PAJARES MARTINSANZ

COLABORADORA EXTERNA
CLARA ISABEL LÓPEZ GÓNZALEZ

GRADO EN INGENIERÍA DEL SOFTWARE
FACULTAD DE INFORMÁTICA
UNIVERSIDAD COMPLUTENSE DE MADRID

Resumen

En este trabajo se propone una aplicación inteligente con capacidad de procesar datos sensoriales procedentes de un dispositivo móvil, del tipo smartphone o tablet mediante técnicas basadas en Aprendizaje Profundo (Deep Learning). La finalidad consiste en la detección de acciones de movimiento humanas para su monitorización.

Para ello se propone el uso de una red neuronal recurrente, cuyo modelo exacto es del tipo LSTM (Long Short Term Memory). Esta red está entrenada con datos de Aceleración, en los tres ejes (X, Y, Z), procedentes de los sensores del smartphone para reconocer cinco tipos concretos de acciones: “Dancing”, “Running”, “Sitting”, “Standing” y “Walking”.

La aplicación está diseñada de tal manera que cuando se reciben datos de aceleración, correspondientes a una de las acciones mencionadas, éstos se procesan en la red para identificar el tipo de acción asociada.

Además, de lo anterior, el trabajo posee una funcionalidad añadida bajo el paradigma de Internet de las Cosas (IoT, Internet of Things), de forma que los datos son enviados a una plataforma específica para IoT su almacenamiento junto con el resultado de su procesamiento mediante la red LSTM. Desde esta plataforma se pueden enviar mensajes sobre las acciones realizadas para ser recibidos en una cuenta de la red social Twitter, a la vez que es posible su visualización accediendo a la mencionada plataforma con fines de monitorización de las acciones almacenadas en ella.

Por tanto, la aplicación desarrollada combina dos capacidades de interés, la primera se refiere a la utilización de técnicas inteligentes avanzadas mediante Deep Learning y la segunda a su extensión bajo el ámbito de IoT.

Los experimentos realizados han verificado la validez de la propuesta, destacando los aspectos relacionados con el sistema inteligente y la navegación autónoma del vehículo autómata.

Palabras clave: Aprendizaje profundo, LSTM, IoT, aceleración, acciones de movimiento, entrenamiento, clasificación.

Abstract

In this work, an intelligent application is proposed with the ability to process sensory data coming from a mobile device, such as a smartphone or tablet, and using techniques based on Deep Learning. The purpose is the detection of human movement actions for monitoring.

A recurrent neural network is proposed for this purpose, whose exact model is of the type of LSTM (Long Short-Term Memory). This network is trained with acceleration data, in the three axes (X, Y, Z), from the smartphone sensors to recognize five specific types of actions: “Dancing”, “Running”, “Sitting”, “Standing” and “Walking”.

The application is designed in such a way that when acceleration data are received, corresponding to one of the mentioned actions, they are processed in the network to identify the type of associated action.

In addition to the above, a functionality, under the paradigm of Internet of Things (IoT, Internet of Things), is added, so that the data are sent to a specific IoT platform for storage together with the result of their processing through the LSTM net. This platform offers a offers a messaging service sending the type of action recognized and stored, which can be received through a Twitter social network account, while it is possible to gain access to the platform for the purposes of monitoring the actions stored in it.

Therefore, the developed application combines two abilities of interest, the first refers to the use of advanced intelligent techniques through Deep Learning and the second to its extension under the scope of IoT.

Keywords: Deep Learning, LSTM, IoT, acceleration, movement actions, training, classification.

Índice

Capítulo 1. Introducción

1.1 Antecedentes

El presente trabajo se enmarca dentro del ámbito de la Inteligencia Artificial (IA), cuya esencia consiste en que una máquina o sistema imite, lo más fielmente posible, comportamientos de los seres humanos, pudiendo así razonar, aprender y tomar decisiones.

Alan Turing fue uno de los pioneros de tratar con la IA, cuando en 1950 (Turing, 1950) publicó su famoso artículo Computing Machinery and Intelligence en el que se planteaba si las máquinas eran capaces de pensar, AI (2022). Desde entonces y hasta el momento actual la evolución de esta disciplina ha sido constante, hasta llegar a los tiempos actuales donde se ha producido una explosión significativa debido, fundamentalmente al desarrollo tecnológico con la aparición de distintas tecnologías y metodologías, que convenientemente combinadas han hecho posible los actuales avances en esta disciplina. Como consecuencia de ello, la IA facilita la vida de las personas en diferentes facetas. Por citar sólo algunas, cabe señalar cómo la utilización de sistemas basados en GPS permite calcular rutas inteligentes teniendo en cuenta las diferentes ubicaciones, a la vez que consideran variables tales como zonas de difícil acceso, con dificultad o de difícil acceso. En el ámbito del reconocimiento de voz o tratamiento de textos son bien conocidos los asistentes inteligentes tales como Siri o Alexa, de forma que interpretan y reconocen el mensaje de voz recibido para realizar una acción determinada, como puede ser activar un dispositivo para poner una canción o apagar un interruptor determinado, acceder a determinados servicios, como los meteorológicos. Los traductores de texto también se encuadran bajo el paradigma de procesamiento del lenguaje natural con carácter inteligente.

Continuando dentro del contexto de las aplicaciones inteligentes, resultan bien conocidos los sistemas de monitorización de actividades físicas realizadas por las personas. Existen aplicaciones instaladas en la mayoría de los dispositivos móviles con capacidad de contar el número de pasos o la distancia recorrida durante determinados espacios de tiempo, tal es el caso de Adidas Running o GoogleFit por citar sólo algunas. Dentro del control de las mencionadas actividades físicas algunos dispositivos, tales como los velocímetros de Garmin (Garmin, 2022) que incluyen sensores de velocidad y cadencia del movimiento que permiten detectar posibles cambios bruscos de movimiento y cadencias de velocidad y enviar avisos a

través de un *smartphone* con conexión *bluetooth* ante posibles incidencias tales como caídas de una bicicleta.

Es justamente en el ámbito de la monitorización física de actividades de personas donde se enmarca el presente trabajo. Más concretamente se propone el desarrollo de un sistema inteligente que identifica la actividad realizada por una persona a partir de los datos de movimiento proporcionados por los sensores embebidos dentro de un dispositivo móvil (*smartphone* o *tablet*). Los datos disponibles son, generalmente: aceleración, velocidad, campo magnético, orientación, velocidad angular y posición. Estos datos se reciben en el propio dispositivo móvil a través de la correspondiente aplicación de Matlab (App Matlab Mobile, 2022), previamente instalada según el sistema operativo de Android o iOS.

Con los valores de aceleración se entrena un modelo de red neuronal específicamente diseñada con tal propósito. Se trata de una red específica dentro de la IA y bajo el paradigma de lo que se conoce como Aprendizaje Profundo (*Deep Learning*) y más concretamente el modelo denominado Long-Short Term Memory (LSTM) con capacidad de procesamiento de secuencias de datos recibidos a lo largo del tiempo, que es justamente como se reciben los valores de aceleración asociados con una acción determinada. El mecanismo de procesamiento de las secuencias temporales, junto con el modelo de esta red se puede encontrar en Goodfellow y col. (2016) o Pajares y col. (2021). De hecho, Sa-nguannarm y col. (2021) recomiendan la utilización de acelerómetros para la identificación de acciones de movimiento mediante técnicas basadas en los modelos LSTM.

De esta manera, se pueden monitorizar determinadas actividades realizadas por una persona equipada con un dispositivo móvil y con la App instalada de Matlab. El modelo de red LSTM se encuentra previamente entrenado con capacidad para identificar las siguientes acciones: “Dancing”, “Running”, “Sitting”, “Standing” y “Walking”. Una vez recibida una secuencia de datos de aceleración el programa indicará el tipo de actividad que corresponda a dichos datos según las categorías anteriores.

Otro aspecto destacable importante es que los resultados del procesamiento y los propios datos pueden ser tratados desde el punto de vista de otro de los paradigmas emergentes actuales, cual es, el conocido como Internet de las Cosas (IoT, *Internet of Things*), que también se aborda en el presente proyecto. En este caso, el dispositivo móvil es la “Cosa” que proporciona los datos, como se ha indicado previamente, los cuales se procesan de forma inteligente mediante el modelo de red LSTM bajo Matlab (2022). Los resultados del

procesamiento se almacenan en la correspondiente plataforma en la nube, en este caso a través de la plataforma ThingSpeak (2022), que es específica de Matlab para IoT.

Con este planteamiento es importante mencionar que este diseño permite su aplicación a diversos campos que, sin duda, pueden ayudar a mejorar la calidad de vida de las personas. Por ejemplo, en el ámbito de la medicina y de cara a procesos de rehabilitación se puede controlar de forma inteligente las actividades realizadas en las diferentes sesiones. O bien, estudiar la causa-efecto de determinados tratamientos farmacológicos en función de la actividad realizada a lo largo del tiempo.

Durante la realización de las actividades bajo monitorización se pueden programar acciones en ThingSpeak para que avise de la realización de las mismas, por ejemplo vía Twitter, a la vez que se almacenan los datos en esta plataforma y se visualizan de forma remota, de forma que se favorece el seguimiento y control de las actividades bajo una supervisión inteligente.

Este trabajo se inspira en modelos y planteamientos previos con la misma o similar finalidad. En este sentido, Bayat y col. (2014) utilizan un *smartphone* también para obtener datos de aceleración, según los tres ejes de coordenadas (X, Y, Z), exactamente como los utilizados en este proyecto. Si bien, como método de identificación de las acciones se utiliza una red neuronal del tipo retro-propagación junto con otras técnicas inteligentes como las Máquinas de Vectores Soporte, que también se utilizan en Pilataxi y col. (2019) con la misma finalidad. Kozina y col. (2013) identifican las caídas de personas por cambios bruscos en el movimiento de un acelerómetro, similar a la funcionalidad que ofrece el dispositivo Garmin (2022) mencionado previamente.

Por otra parte, en los dos cursos precedentes, tanto Pavón (2020) como Herrero-Fernández y Jiménez-González (2021) desarrollaron sendas aplicaciones, en sus correspondientes trabajos fin de grado, con similares características a la propuesta en el presente trabajo y por tanto, dentro del paradigma de Aprendizaje Profundo. De hecho, en ambos casos se utilizaron también como datos los valores de aceleración en los tres ejes y el modelo de red LSTM. En el primer caso el procesamiento de datos se realiza en un servidor adaptado al efecto y en el segundo caso, el tratamiento se realiza en modo local, al igual que en este trabajo. Además, en Herrero-Fernández y Jiménez-González (2021) también se utiliza IoT y la plataforma ThingSpeak para recepción y monitorización de los datos.

1.2 Objetivos

El principal objetivo general que se plantea en este trabajo consiste en identificar los cinco tipos de acciones de movimiento indicadas previamente, mediante un modelo de red neuronal del tipo LSTM en el ámbito de las redes neuronales recurrentes (RNN). Las acciones son realizadas por una persona según los datos de aceleración que capturan los sensores, previa activación, de un dispositivo móvil (Android o iOS).

Como objetivo secundario, se propone la monitorización de estos movimientos a través del acceso a la plataforma ThingSpeak como base del paradigma IoT. Todo ello, integrado en una aplicación con una interfaz sencilla e intuitiva.

Para la consecución de estos objetivos generales se proponen los siguientes objetivos específicos:

- Configurar el dispositivo sensorial para la captura de datos.
- Diseño de la funcionalidad para la recepción de los datos de aceleración y su envío a la red LSTM entrenada para la clasificación de la acción.
- Configuración de la plataforma IoT en ThingSpeak.
- Integrar todas las funcionalidades anteriores y establecer los mecanismos de clasificación.
- Diseño de la interfaz de usuario para manejo de la aplicación con el fin de capturar, procesar, visualizar y monitorizar las acciones realizadas.

1.3 Plan de Trabajo

El plan de trabajo se estructura considerando el mismo número de tareas que objetivos, coincidiendo exactamente con los objetivos indicados anteriormente y con una distribución equitativa temporal a lo largo del curso. En cualquier caso, se concretan como sigue:

- Revisión y estudio de aplicaciones relacionadas con esta propuesta.
- Estudio y ampliación de los conocimientos previos sobre el lenguaje de programación elegido, en este caso Matlab y sus herramientas asociadas, como es la denominada App Designer para el desarrollo de la aplicación.
- Adaptar y configurar el dispositivo para la captura de datos de movimiento, y en concreto de aceleración.

- Estudio y aprendizaje de los métodos de procesamiento de secuencias de datos según el modelo de red LSTM.
- Configuración del modelo de red LSTM.
- Estudio, configuración, programación y manejo de la plataforma ThingSpeak con el fin de almacenar las acciones de movimiento identificadas por la red LSTM.
- Diseño de las unidades funcionales que constituyen el conjunto de la aplicación.
- Diseño de la interfaz de usuario para la integración de las mencionadas funcionalidades.
- Análisis del funcionamiento de la aplicación en su conjunto.
- Elaboración de la memoria, incluyendo las correspondientes revisiones y modificaciones.

1.4 Organización de la memoria

La memoria se estructura en capítulos de la siguiente manera:

- El primer capítulo, que es la introducción, además de los aspectos preliminares, describe los objetivos y planificación del proyecto.
- El segundo capítulo explica los distintos tipos de redes y su funcionamiento.
- El tercer capítulo describe el diseño de la aplicación, comenzando con la descripción de las herramientas utilizadas y la arquitectura de misma.
- El cuarto capítulo expone los resultados obtenidos, tanto desde el punto de vista de entrenamiento, como de clasificación y el manejo de las funcionalidades en la nube.
- El quinto capítulo expone las conclusiones y las posibles mejoras futuras de la aplicación.

Chapter 1. Introduction

1.1 Preliminary

This work belongs to the field of Artificial Intelligence (AI), whose essence is that a machine or system imitates, as closely as possible, the behavior of human beings, thus being able to reason, learn and make decisions.

Alan Turing was one of the pioneers of dealing with AI, when in 1950 (Turing, 1950) he published his famous article Computing Machinery and Intelligence in which he wondered whether machines were capable of thinking, AI (2022). Since then and until now, the evolution of this discipline has been constant, until reaching the current times where there has been a significant explosion, mainly due to technological development with the appearance of different technologies and methodologies, which conveniently combined have made possible current advances in this discipline. Consequently, AI facilitates people's lives in different facets. To mention just a few, it should be noted how the use of GPS-based systems allows the calculation of intelligent routes taking into account different locations, while considering variables such as areas that are difficult to access, difficult to reach or difficult to access. In the field of speech recognition or text processing, intelligent assistants such as Siri or Alexa are well known, so that they interpret and recognize the voice message received to perform a specific action, such as activating a device to play a song or turn off a certain switch, access certain services, such as weather. Text translators also are part of the paradigm of intelligent natural language processing.

Continuing in the context of intelligent applications, systems for monitoring the physical activities performed by people are well known. There are applications installed in most mobile devices with the capacity to count the number of steps or the distance traveled during certain periods of time, such as Adidas Running or GoogleFit, to cite just a few. Within the control of the aforementioned physical activities some devices, such as Garmin speedometers (Garmin, 2022) that include speed and cadence of movement sensors that can detect possible sudden changes in movement and speed cadences and send warnings through a smartphone with Bluetooth connection to possible incidents such as falling off a bicycle.

It is precisely in the field of physical monitoring of people's activities where the present work is framed. More specifically, we propose the development of an intelligent system that identifies the activity carried out by a person from the movement data provided by the sensors

embedded in a mobile device (smartphone or tablet). The available data are generally: acceleration, velocity, magnetic field, orientation, angular velocity and position. These data are received in the mobile device itself through the corresponding Matlab application (Matlab Mobile App, 2022), previously installed according to the Android or iOS operating system.

A neural network model specifically designed for this purpose is trained with the acceleration values. It is a specific network within AI and under the paradigm of what is known as Deep Learning and more specifically the model called Long-Short Term Memory (LSTM) with the capacity to process data sequences received over time, which is precisely how the acceleration values associated with a given action are received. The mechanism of temporal sequence processing, along with the model of this network can be found in Goodfellow et al. (2016) or Pajares et al. (2021). In fact, Sanguannarm et al. (2021) recommend the use of accelerometers for the identification of movement actions using techniques based on LSTM models.

In this way, it is possible to monitor certain activities performed by a person equipped with a mobile device and the installed Matlab App. The LSTM network model is previously trained with the ability to identify the following actions: "Dancing", "Running", "Sitting", "Standing" and "Walking". Once a sequence of acceleration data is received, the program will indicate the type of activity that corresponds to the data according to the above categories.

Another important aspect is that the results of the processing and the data itself can be treated from the point of view of another of the current emerging paradigms, which is known as the Internet of Things (IoT, Internet of Things), which is also covered in this project. In this case, the mobile device is the "Thing" that provides the data, as previously indicated, which are processed intelligently by the LSTM network model under Matlab (2022). The processing results are stored in the corresponding cloud platform, in this case through the ThingSpeak (2022) platform, which is specific to Matlab for IoT.

With this idea in mind, it is important to mention that this design allows its application to various fields that, without a doubt, can help improve people's quality of life. For example, in the field of medicine and for rehabilitation processes, the activities carried out in the different sessions can be intelligently controlled. Or study the cause-effect of certain pharmacological treatments based on the activity carried out over time.

During the performance of the activities under monitoring, actions can be programmed in ThingSpeak so that it notifies of their performance, for example via Twitter, while the data is

stored on this platform and viewed remotely, so that monitoring and control of activities under intelligent supervision is favored.

This work is inspired by previous models and approaches with the same or similar purpose. In this sense, Bayat et al. (2014) also uses a smartphone to obtain acceleration data, according to the three coordinate axes (X, Y, Z), exactly like the ones used in this project. Although, as a method of identifying actions, a back-propagation type neural network is used together with other intelligent techniques such as Support Vector Machines, which are also used in Pilataxi et al. (2019) with the same goal. Kozina et al. (2013) identify the falls of people due to sudden changes in the movement of an accelerometer, similar to the functionality offered by the previously mentioned Garmin (2022) device.

On the other hand, in the two preceding courses, both Pavón (2020) and Herrero-Fernández and Jiménez-González (2021) developed applications, in their corresponding thesis, with similar characteristics to the one proposed in the present work and, therefore, within the Deep Learning paradigm. In fact, in both cases the acceleration values in the three axes and the LSTM network model were also used as data. In the first case, the data processing is performed on a server adapted for this purpose and in the second case, the processing is performed in local mode, as in this work. In addition, in Herrero-Fernández and Jiménez-González (2021), IoT and the ThingSpeak platform are also used for data reception and monitoring.

1.2 Objectives

The main general objective of this work is to identify the five types of movement actions previously indicated, through a neural network model of the LSTM type in the field of recurrent neural networks (RNN). The actions are performed by a person according to the acceleration data captured by sensors, prior activation, from a mobile device (Android or iOS).

As a secondary objective, the monitoring of these movements is proposed through access to the ThingSpeak platform as the basis of the IoT paradigm. All of that, integrated in an application with a simple and intuitive interface.

In order to achieve these general objectives, the following specific objectives are proposed:

- Configuring the sensing device for data capture.

- Designing the functionality for receiving the acceleration data and sending it to the trained LSTM network for action classification.
- Configuration of the IoT platform in ThingSpeak.
- Integrating all the above functionalities and setting up the classification mechanisms.
- Designing the user interface for handling the application in order to capture, process, visualize and monitor the actions performed.

1.3 Workplan

The work plan is structured considering the same number of tasks as objectives, coinciding exactly with the objectives indicated above and with an equal distribution in time throughout the course. In any case, they are specified as follows:

- Review and study of applications related to this proposal.
- Study and expansion of previous knowledge about the chosen programming language, in this case Matlab and its associated tools, such as the so-called App Designer for the development of the application.
- Adapt and configure the device for the capture of motion data, and specifically acceleration.
- Study and learning of the data sequence processing methods according to the LSTM network model.
- Configuration of the LSTM network model.
- Study, configuration, programming and management of the ThingSpeak platform in order to store the motion actions identified by the LSTM network.
- Design of the functional units that constitute the whole application.
- Design of the user interface for the integration of the aforementioned functionalities.
- Analysis of the operation of the application as a complete set.
- Preparation of the memory, including the corresponding revisions and modifications.

1.4 Report organization

The report is structured in chapters as follows:

- The first chapter, which is the introduction, in addition to preliminary aspects, describes the objectives and planning of the project.
- The second chapter explains the different types of networks and their functioning.
- The third chapter describes the design of the application, starting with the description of the tools used and its architecture.
- The fourth chapter presents the results obtained, both from the point of view of training, classification and management of the functionalities in the cloud.
- The fifth chapter presents the conclusions and possible future improvements of the application.

Capítulo 2. Modelos de redes

2.1 Introducción

El presente capítulo describe el modelo de red neuronal aplicado en este trabajo, para el que se sigue la referencia de Pajares y col. (2021). Concretamente, las Redes Neuronales Recurrentes (RNN, *Recurrent Neural Network*) y bajo el concepto de ellas el modelo conocido como *Long Short-Term Memory* (LSTM).

Como se especifica más adelante, la aplicación recibe secuencias de entrada temporales de la forma $\left(\boldsymbol{x}_t, \boldsymbol{y}_t\right)_{t=1}^{T}$. En este caso concreto son secuencias de movimientos a partir de los datos suministrados por un *smartphone*, figura 2.1, dotado de tecnologías sensoriales apropiadas, que proporcionan los siguientes datos secuenciados en el tiempo:

1) *Orientación*, expresada en grados con respecto a los correspondientes ejes *X, Y* y *Z*, esto es: *azimut* (*yaw*, ψ) o ángulo entre el eje *Y* positivo y el norte magnético *N*, definido en el rango [0°,360°]; *cabeceo* (*pitch*, θ) positivo o giro del eje *Z* positivo hacia el eje *Y* positivo; y *alabeo* (*roll*, ϕ) positivo o giro del eje *Z* positivo hacia el eje *X* positivo.
2) *Velocidad angular* (w_x, w_y, w_z) en los respectivos ejes *X*, *Y* y *Z* en *rad*/*s*;
3) *Aceleración* (a_x, a_y, a_z) en los respectivos ejes *X*, *Y* y *Z* en m/s^2.

En cada instante de tiempo *t* se puede definir un vector de entrada a la red que caracteriza las aceleraciones, tal como: $\boldsymbol{x}_t \equiv (a_x, a_y, a_z)_t \in \Re^3$ de forma que a cada secuencia se le asocia una etiqueta que en realidad es una clase $\boldsymbol{y}_t \equiv c_i$, siendo *i* el número de clases definidas. De esta forma, el objetivo consiste en determinar el estado de movimiento de una persona equipada con un dispositivo móvil según la actividad que esté desarrollando en cada momento concreto, por ejemplo, entre otras, $c_1 \equiv$ sentada; $c_2 \equiv$ saltando; $c_3 \equiv$ corriendo o $c_4 \equiv$ caminando. En este caso se trataría de un sistema con cuatro clases y por tanto *i* es cuatro en este ejemplo. El vector $\boldsymbol{x}_t$ puede definirse con los nueve valores, tres de orientación, tres de velocidad angular y otros tres de aceleración, de forma que en este caso $\boldsymbol{x}_t \in \Re^9$.

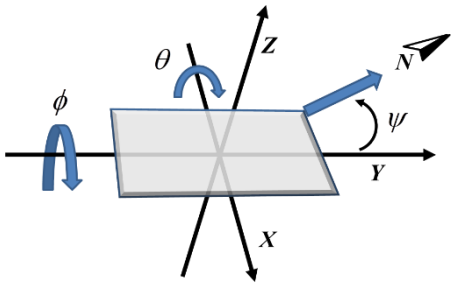


Figura 2.1 Ángulos de orientación

Esta es la idea expuesta en el trabajo de Rego-Drumond y col. (2018), donde los valores de orientación, velocidad angular y aceleración se obtienen mediante dispositivos del tipo IMU (*Inertial Measurement Units*), esto es, unidades de medición inercial. A modo de ejemplo ilustrativo, las cuatro clases definidas previamente se representan por los siguientes valores: c_1 ≡ (1,0,0,0), c_2 ≡ (0,1,0,0), c_3 ≡ (0,0,1,0) y c_4 ≡ (0,0,0,1).

2.2 Redes Neuronales Recurrentes

Las RNN se caracterizan por su utilidad para resolver problemas donde aparecen datos secuenciales, de forma que pueden retener información de un estado en la secuencia de una iteración a la siguiente (Rumelhart y col., 1986). La clave de estos modelos está en el hecho de que comparten parámetros, lo que permite su aplicación a problemas de distinta naturaleza, a la vez que se consiguen generalizaciones específicas para extraer la información procedente de los datos.

Conviene señalar que, generalmente, las secuencias están estructuradas en el tiempo, siendo por tanto el tiempo una de las variables clave en estos procesos. Justamente esto es lo que sucede en el presente trabajo, de forma que los datos a procesar son componentes de aceleración, que representan acciones de movimiento.

Existen diversos modelos de RNN, cuya representación más simple es la que se muestra en la figura 2.2 tanto en su versión plegada como desplegada.

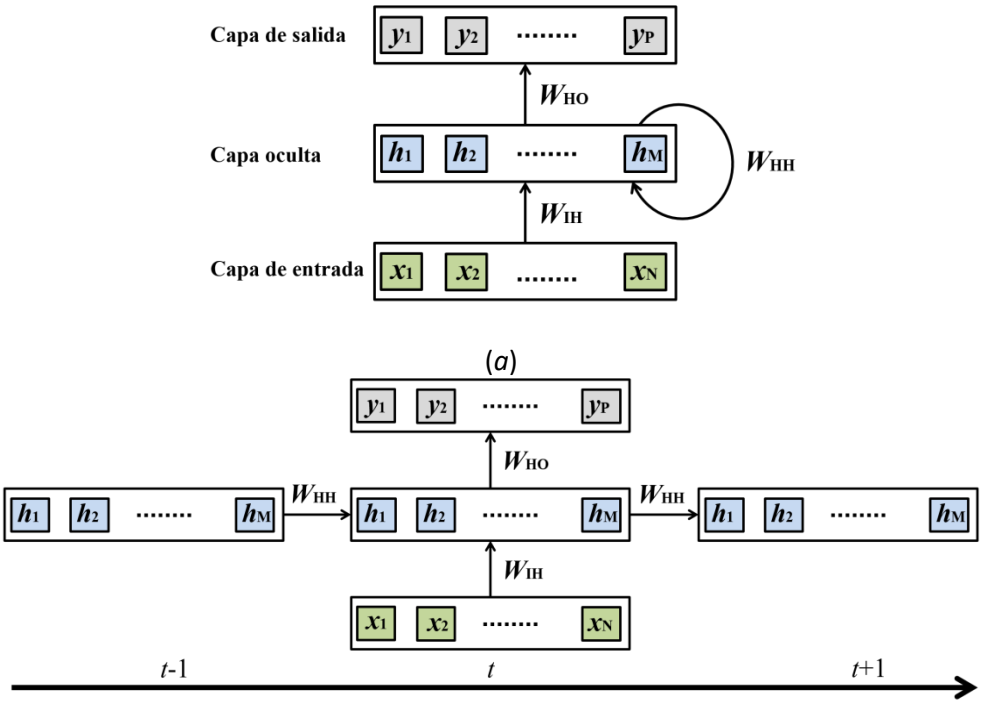


Figura 2.2 Modelo de red RNN: (a) plegada y (b) desplegada

Cada flecha representa una conexión total entre las capas (*fully connected*), que se establece mediante los correspondientes pesos representados por las matrices ***W***, que son justamente las estructuras para aprender durante el proceso de entrenamiento de la red. Existen conexiones laterales y verticales. Las verticales conectan las respectivas capas de entrada, oculta y salida en un instante de tiempo y por tanto para una entrada dada ***x***. En el caso que nos ocupa estos vectores ***x*** son las componentes de aceleración proporcionadas por el dispositivo móvil en el instante de tiempo ***t***, mientras que las laterales permiten conexiones entre distintos instantes de tiempo (*t*-1, *t*, *t*+1).

En la figura 2.3 se muestra la arquitectura del modelo de forma extendida y en un mayor nivel de detalle. Así, para cada entrada $\boldsymbol{x}_t$ se genera una salida $\boldsymbol{y}_t$, esto es, para una entrada con las tres componentes de aceleración se produce una salida que representa la acción realizada.

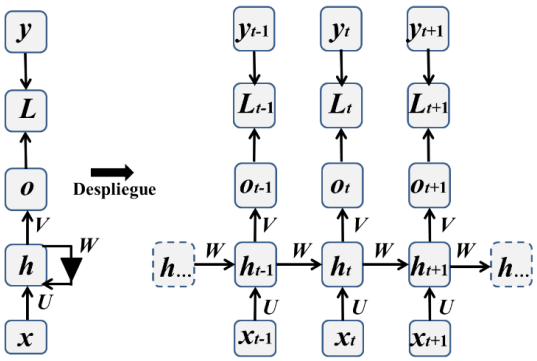


Figura 2.3 Redes recurrentes con salidas y conexiones entre unidades ocultas

El mecanismo de funcionamiento de este modelo se puede sintetizar como sigue:

a) La secuencia de entrada $\boldsymbol{x}_t$ genera la secuencia de salida $\boldsymbol{o}_t$. La función de pérdida L mide la similitud de $\boldsymbol{o}_t$ con respecto a la salida objetivo o deseada $\boldsymbol{y}_t$, de forma que internamente se obtiene $\hat{\boldsymbol{y}}_t = softmax(\boldsymbol{o}_t)$ comparándola con $\boldsymbol{y}_t$.

b) El modelo RNN tiene entradas a conexiones ocultas definidas por la matriz de pesos $\boldsymbol{U}$, conexiones recurrentes entre unidades ocultas parametrizadas por una matriz de pesos $\boldsymbol{W}$ y conexiones entre unidades ocultas hacia la salida, parametrizadas por una matriz de pesos $\boldsymbol{V}$.

c) La función total de pérdida para una secuencia dada de valores $\boldsymbol{x}$ emparejados con una secuencia de valores de salida $\boldsymbol{y}$, es justo la suma de las funciones de pérdida sobre todos los pasos de tiempo. La pérdida total para una secuencia dada de valores $\boldsymbol{x}$ emparejados con una secuencia de valores $\boldsymbol{y}$ sería exactamente la suma sobre todos los pasos de tiempo. Por ejemplo, si L_t es la estima de máxima verosimilitud (*log-likelihood*) negativa de y_t dadas $\boldsymbol{x}_1,\ldots,\boldsymbol{x}_t$, entonces,

$$L\left(\{\boldsymbol{x}_1,\ldots,\boldsymbol{x}_t\},\{\boldsymbol{y}_1,\ldots,\boldsymbol{y}_t\}\right)=\sum_t L_t=-\sum_t log\ p_{modelo}\left(y_t\,|\{\boldsymbol{x}_1,\ldots,\boldsymbol{x}_t\}\right) \qquad (2.1)$$

donde $p_{modelo}\left(y_t\,|\{\boldsymbol{x}_1,\ldots,\boldsymbol{x}_t\}\right)$ viene determinado leyendo la entrada para $\boldsymbol{y}_t$ a partir del vector modelo $\hat{\boldsymbol{y}}_t$.

d) Para el entrenamiento se requiere un número considerable de datos, con sus correspondientes asociaciones entrada-salida, es decir los mencionados pares ($\boldsymbol{x}_t$, $\boldsymbol{y}_t$) que asocian los datos de aceleración con la acción realizada.

Por otra parte, existen diferentes topologías de red teniendo en cuenta la conexión entre las distintas capas Goodfellow y col. (2016). En la figura 2.4 se muestran varias estructuras

topológicas. Para cada uno de ellos se muestran algunos ejemplos ilustrativos de aplicación según el esquema correspondiente:

- *Uno a uno*: un objeto de una imagen → una etiqueta de su clasificación.
- *Uno a muchos*: imagen → sentencia descriptiva de la imagen.
- *Muchos a uno*: frase → sentimiento (positivo o negativo).
- *Muchos a muchos* (secuencia-secuencia): datos de aceleración → acción de movimiento.
- *Muchos a muchos* (secuencia-secuencia sincronizada): *frames* de vídeo → *bounding boxes* asociados con cada objeto.

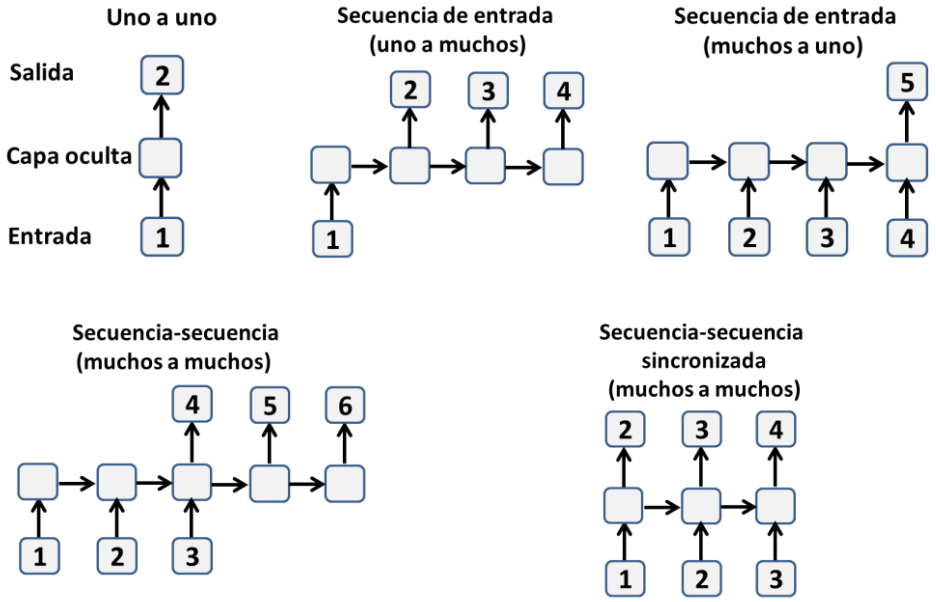


Figura 2.4 Diferentes topologías de red

Como puede observarse, el modelo que se corresponde con este trabajo es el "muchos a muchos sincronizado", ya que las secuencias de entrada $\boldsymbol{x}_t$ se producen espaciadas en el tiempo, esperando para esa misma entrada y en el mismo instante de tiempo una salida, $\boldsymbol{y}_t$, de aquí el concepto de sincronización.

2.3 Redes LSTM (Long Short-Term Memory)

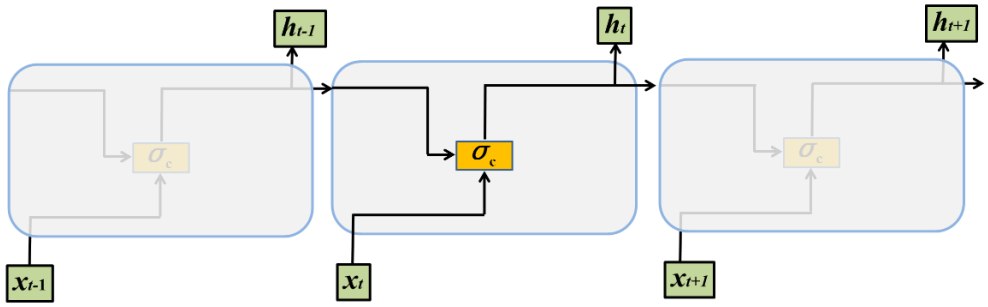
Resulta bien conocido que en las redes neuronales profundas las dependencias conocidas como *long-term* constituyen un reto importante. El problema se resume en el hecho de que los gradientes que se retropropagan sobre muchos estados tienden o bien a desaparecer, la

mayoría de las veces, o a una explosión computacional, menos a menudo, que afecta claramente al proceso de optimización. Esto se debe principalmente a que el gradiente depende de los errores, tanto actuales como pasados, de forma que la acumulación continuada de errores origina problemas importantes para memorizar dependencias a largo alcance, de aquí la terminología denominada *long-term*. Goodfellow y col. (2016) proporcionan algunas propuestas para resolver esta problemática. Dentro de las citadas propuestas destaca la que se describe a continuación, que se basa en el concepto denominado *Long Short-Term Memory* (LSTM) que permite especificar la información que debe preservarse o eliminarse, esto es, almacenarse en forma de memoria, de ahí el término *memory*, o descartarse. Este es el modelo propuesto en este trabajo. Como en cualquier modelo de aprendizaje, como es el caso, estos modelos constan de dos fases: Aprendizaje y Clasificación.

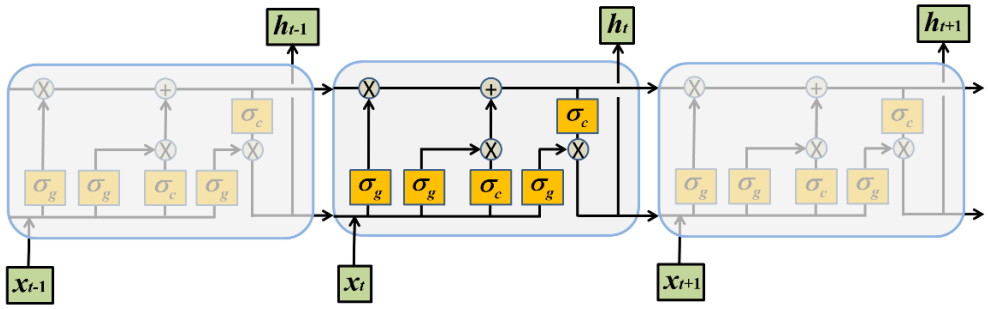
2.3.1 Entrenamiento y aprendizaje

Las LSTM fueron introducidas por Hochreiter y Schmidhuber (1997) con el fin de resolver el problema del gradiente indicado previamente. Se trata de una red RNN específica que procesa una secuencia de pares de entrada-salida $\left(\boldsymbol{x}_t, \boldsymbol{y}_t\right)_{t=1}^{T}$. Para cada par de valores $\left(\boldsymbol{x}_t, \boldsymbol{y}_t\right)$, la correspondiente celda LSTM toma una nueva entrada $\boldsymbol{x}_t$ y el valor oculto $\boldsymbol{h}_{t\text{-}1}$ obtenido en el paso previo, y con ello produce una estima $\hat{\boldsymbol{y}}_t$ para la salida objetivo $\boldsymbol{y}_i$ establecida al efecto y dada la secuencia de entrada previa $\boldsymbol{x}_1$, $\boldsymbol{x}_2$,…, $\boldsymbol{x}_t$ también con un nuevo valor oculto $\boldsymbol{h}_t$ y un nuevo valor del estado de la celda o memoria $\boldsymbol{C}_t$. La figura 2.5 muestra de forma detallada la estructura de una celda LSTM sobre el tiempo, con el cómputo establecido como se describe seguidamente tal y como aparece en Graves (2013).

En la referida figura se muestra una estructura característica del mencionado tipo LSTM tal y como se especifica en los trabajos de Gers y col. (2002) y Olah (2015). Cada celda temporal recibe, en cada paso de tiempo, una muestra de la secuencia $\boldsymbol{x}$ procedente de la capa de entrada, y envía el estado de la celda actualizado, así como el valor de la salida $\boldsymbol{h}$ al siguiente paso, es decir, a la siguiente celda. Por otra parte, en cada paso también se envía el correspondiente valor $\boldsymbol{h}$ a la capa de salida. Estos términos están indexados por el subíndice *t* correspondiente que establecen los pasos concretos de la secuencia en cada una de las distintas unidades que conforman el modelo.



(*a*)



(*b*)

Figura 2.5 Estructura general y conexión de las celdas LSTM

El estado de cada celda viene a ser una línea de transporte que atraviesa la celda con las interacciones que se indican explícitamente en la figura 2.6, de modo que la LSTM puede eliminar o añadir información al estado de la celda mediante estructuras que se denominan en la terminología especializada como *gates*, formadas por operaciones del tipo mostrado en la subfigura de la parte derecha de la figura 2.5 con el flujo de información indicado por las flechas en dicha subfigura.

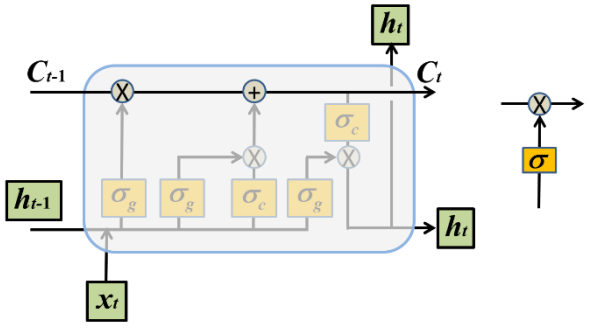


Figura 2.6 Línea de estado de la celda

El primer paso en una estructura LSTM es decidir qué información eliminar del estado de la celda, cuya responsable es una capa sigmoidal denominada puerta de olvido o *forget gate*, $\boldsymbol{f}_t$, definida según la ecuación (2-2) y la figura 2.7,

$$\boldsymbol{f}_t = \sigma_g\left(U_{xf}\,\boldsymbol{x}_t + W_{hf}\,\boldsymbol{h}_{t-1} + \boldsymbol{b}_f\right) \tag{2.2}$$

de forma que teniendo en cuenta los pesos U_{xf} y W_{hf} y un *bias* $\boldsymbol{b}_f$, a través de la función *sigmoide* (σ_g) se genera un valor entre 0 y 1 para cada estado de la celda $\boldsymbol{C}_{t-1}$. Un valor de 1 significa mantener este estado por completo y un 0 deshacerse de él. Por ejemplo, en el modelo de identificación de acciones de movimiento, que trata de predecir la siguiente acción en base a todas las anteriores, el estado de la celda podría incluir la acción actual. Cuando se ve un elemento nuevo, la idea podría ser conservar la acción anterior por considerar que los movimientos entre transiciones son suaves y no bruscas. Es decir, si estamos realizando la acción de correr, el paso a una situación de sentado, no es inmediato, requiriendo un proceso de transición.

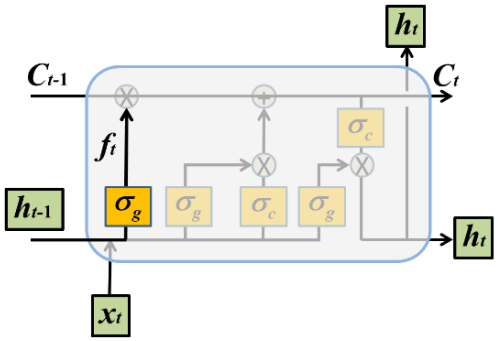


Figura 2.7 Forget gate

El siguiente paso consiste en decidir qué información nueva se va a almacenar en el estado de la celda, lo cual tiene dos partes. Primero, una capa *sigmoide* llamada puerta de entrada (*input gate*) $\boldsymbol{i}_t$, figura 2.8, decide qué valores se actualizan.

$$\begin{aligned}\boldsymbol{i}_t &= \sigma_g\left(U_{xi}\,\boldsymbol{x}_t + W_{hi}\,\boldsymbol{h}_{t-1} + \boldsymbol{b}_i\right)\\ \boldsymbol{g}_t &= \sigma_c\left(U_{xc}\,\boldsymbol{x}_t + W_{hc}\,\boldsymbol{h}_{t-1} + \boldsymbol{b}_c\right)\end{aligned} \tag{2.3}$$

Seguidamente, una capa σ_c (generalmente de tipo tangente hiperbólica, *tanh*) crea nuevos valores candidatos $\boldsymbol{g}_t$, que podrían agregarse al estado, teniendo en cuenta los pesos U_{xi}, W_{hi}, U_{xc} y W_{hc}, junto con los correspondientes *bias* $\boldsymbol{b}_i$ y $\boldsymbol{b}_c$. En el siguiente paso, se combinan ambos para generar una actualización del estado.

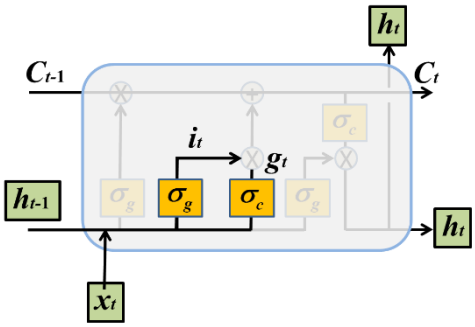


Figura 2.8 Input gate y estado de la celda

Ahora es el momento de actualizar el estado previo de la celda, g_{t-1}, al nuevo estado de esa celda g_t. Los pasos anteriores ya decidieron qué hacer, solo es necesario hacerlo ahora, para ello se multiplica el estado antiguo por f_t, olvidando las cosas que se decidieron olvidar antes. Luego se suma el término i_t e g_t, figura 2.9, que son los nuevos valores candidatos, en función de cuánto se decide actualizar cada valor de estado.

$$C_t = f_t \text{ e } C_{t-1} + i_t \text{ e } g_t \qquad (2.4)$$

En el caso del modelo de detección de acciones, aquí es donde realmente se conserva o se elimina la información sobre la acción anterior anterior y se agrega la nueva información, tal como se decidió en los pasos anteriores.

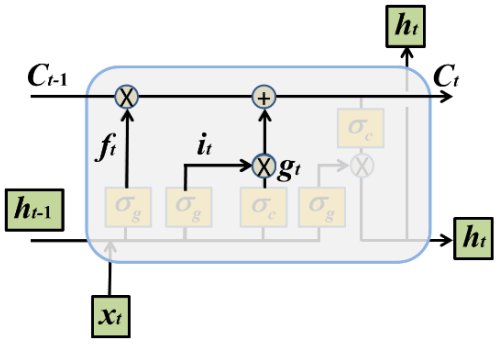


Figura 2.9 Actualización del estado de la celda

Finalmente, es necesario obtener el resultado de la salida, que se basa en el estado de la celda, si bien, se trata de una versión filtrada, figura 2.10. Primero, se aplica la función *sigmoide* que decide qué partes del estado de la celda van a contribuir a la salida. Luego, se aplica la función *tanh*, que sitúa los valores en el rango −1 y 1 y se multiplica por la salida de la puerta de tipo sigmoidal, de modo que solo contribuyen a la salida las partes seleccionadas.

$$o_t = \sigma_g \left(U_{xo} x_t + W_{ho} h_{t-1} + b_o \right) \qquad (2.5)$$
$$h_t = o_t \text{ e } \sigma_c \left(C_t \right)$$

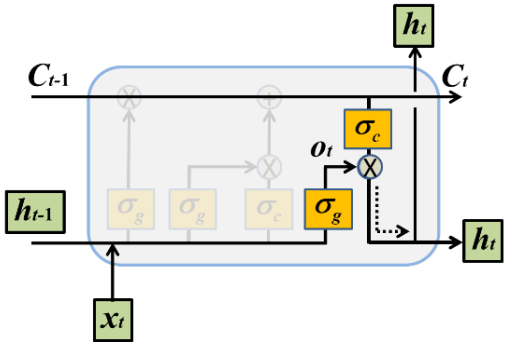


Figura 2.10 Generación de la salida

Por tanto, teniendo en cuenta las relaciones anteriores, la simbología asociada se interpreta como sigue:

1) $\boldsymbol{x}_t \in \Re^d$ representa la entrada a la celda dada, con *d* siendo su dimensionalidad, que se corresponde con el número de características de entrada.

2) $\boldsymbol{h}_t \in \Re^h$ representa el estado *short-term* de la celda, que es igual a la salida $\boldsymbol{y}_t$ de la celda. Su dimensión es *h* que se corresponde con el número de unidades ocultas de la red. Su valor inicial se establece a cero**.**

3) $\boldsymbol{f}_t \in \Re^h$, $\boldsymbol{i}_t \in \Re^h$, $\boldsymbol{g}_t \in \Re^h$, $\boldsymbol{o}_t \in \Re^h$ denotan las puertas de olvido (*forget gate*), entrada (*input*), celda candidata (*cell candidate*) y salida (*output*) respectivamente.

4) El estado de la celda, en el instante de tiempo *t*, viene dado por $\boldsymbol{C}_t = \boldsymbol{f}_t \text{ e } \boldsymbol{C}_{t-1} + \boldsymbol{i}_t \text{ e } \boldsymbol{g}_t$, donde e expresa el producto de Hadamard, esto es la multiplicación de vectores elemento a elemento, con la dimensión apropiada, siendo en este caso $\boldsymbol{C}_t$ de la misma dimensión que sus componentes, esto es $\Re^h$.

5) Las matrices de los pesos de entrada de la red asociados con la entrada $\boldsymbol{x}_t$ tienen las dimensiones siguientes, $U_{xf}, U_{xi}, U_{xc}, U_{xo} \in \Re^{h \times d}$, así como los pesos recurrentes $W_{hf}, W_{hi}, W_{hc}, W_{ho} \in \Re^{h \times h}$ y los *bias* $\boldsymbol{b}_f, \boldsymbol{b}_i, \boldsymbol{b}_c, \boldsymbol{b}_o \in \Re^h$ que en determinadas implementaciones se concatenan como sigue en una única matriz.

$$U = \begin{bmatrix} U_{xi} \\ U_{xf} \\ U_{xc} \\ U_{xo} \end{bmatrix} \in \Re^{4h \times d} \qquad W = \begin{bmatrix} W_{hi} \\ W_{hf} \\ W_{hc} \\ W_{ho} \end{bmatrix} \in \Re^{4h \times h} \qquad \boldsymbol{b} = \begin{bmatrix} \boldsymbol{b}_i \\ \boldsymbol{b}_f \\ \boldsymbol{b}_c \\ \boldsymbol{b}_o \end{bmatrix} \in \Re^{4h \times 1} \qquad (2.6)$$

2.3.2 Parámetros de aprendizaje

Durante el proceso de entrenamiento es necesario definir los parámetros necesarios para el ajuste de los pesos involucrados en el modelo, dentro de ellos destacan por su relevancia los siguientes:

Gradiente descendente, que tiene como objetivo minimizar el error cometido entre una entrada dada y la esperada. Una de las técnicas más habituales es la denominada *Gradiente Descendente Estocástico* (SGD*, Stochastic Gradient Descent*) consiste en minimizar una función objetivo, función de error o *loss* function (función de pérdida) que tiene la forma (Bishop, 2006; Ruder, 2017),

$$J(w)=\frac{1}{n}\sum_{i=1}^{n} J_i(w) \tag{2.7}$$

El argumento de la función, *w*, que minimiza *J*(*w*) debe ajustarse durante el proceso de aprendizaje. El índice *i* en *J* hace referencia a la i-ésima observación, en nuestro caso secuencias de movimiento. Con ello, se está en disposición de minimizar la función de forma iterativa con iteraciones *k*,

$$w(k+1) = w(k) - \varepsilon \frac{1}{n}\sum_{i=1}^{n} \nabla J_i(w) \tag{2.8}$$

Batch size: durante el proceso de actualización de los pesos por minimización de la función de pérdida, tal y como se ha comentado previamente, se proporciona un conjunto de datos que constituye lo que se conoce como lote (*batch*)

Epochs e iteraciones: define el número de veces que el conjunto total de muestras son procesadas en cada paso de la optimización. Los pesos del modelo se actualizan tras el procesamiento de cada *epoch*. Si se establece un número máximo de *epochs*, el proceso de entrenamiento finaliza cuando se alcanza dicho número, pudiendo ocurrir que no se llegue a alcanzar la convergencia del modelo, es decir el mínimo absoluto de la función de error.

Razón de aprendizaje (*learning rate*): determina la rapidez con la que son actualizados los pesos. Cabe la posibilidad, como es el caso, de establecer una razón de aprendizaje variable que se modifica disminuyendo su valor cada cierto número de *epochs*. Para ello se aplica lo que se conoce como factor de reducción.

Método de optimización: es el método concreto utilizado aplicado para minimizar la función de pérdida.

Softmax: es una función que genera valores de salida normalizados en el rango [0,1]. Es decir, calcula la probabilidad de pertenencia de cada muestra de entrada a una de las clases previstas a la salida. Está definida como sigue,

$$softmax(x)_i = \frac{exp(x_i)}{\sum_{j=1}^{n} exp(x_j)} \tag{2.9}$$

2.4 Clasificación

Tras la fase de entrenamiento y con los pesos actualizados, el procedimiento de clasificación consiste en proporcionar nuevos valores de aceleración de entrada, representados por ***x*** procedentes del sensor del dispositivo móvil. Esta entrada se suministra a la red entrenada, generando la correspondiente salida, ***y*** que se clasifica según una de las acciones previstas.

Una vez realizada la clasificación de cada dato, se puede proceder a su incorporación al modelo para realizar un nuevo proceso de entrenamiento con las muestras suministradas tras la clasificación.

Capítulo 3. Diseño de la aplicación

El presente capítulo contiene, en primer lugar, la descripción de las herramientas utilizadas para el desarrollo de la aplicación, la cual se detalla seguidamente.

3.1 Herramientas

Las herramientas tecnológicas utilizadas son concretamente Matlab (2022) y ThingSpeak (2022), que se describen brevemente a continuación.

3.1.1 Matlab

Se han utilizado las versiones R2021b y R2022a de Matlab, haciendo uso de la licencia Campus Wide Suite que proporciona la Universidad Complutense para estudiantes, profesores e investigadores.

Matlab (*Matrix Laboratory*), producto de MathWorks, es un entorno de programación específicamente diseñado para cálculos numéricos con vectores y matrices. Dentro de la IA, puede ser usado para el aprendizaje automático, aprendizaje profundo y aprendizaje por refuerzo, ya que dispone de potentes *toolboxes* para tal propósito, así como acceso a un foro comunitario para intercambio de ejemplos, experimentos y planteamiento de dudas, con una excelente documentación sobre las funcionalidades proporcionadas y un soporte técnico importante. Sus características principales son las que se describen a continuación (Matlab, 2022 y Casado-Fernández, 2022):

- Es un lenguaje de alto nivel.
- Se pueden visualizar datos en gráficas y diagramas, así como gestionar polinomios y funciones entre otros.
- Se pueden simular procesos y sistemas.
- Se integra fácilmente con otros lenguajes como pueden ser C/C++, Java y .NET.
- Se pueden crear aplicaciones de cliente utilizando la herramienta gráfica Matlab App *Designer*.
- Si se precisa de más funcionalidad, se pueden descargar toolboxes que la aporten.

Para la elaboración de este proyecto se ha utilizado concretamente las siguientes utilidades: Matlab Drive, Matlab Desktop, Matlab Mobile y Matlab Online.

Con Matlab Drive, gracias a MathWorks Cloud, se puede acceder a los archivos del proyecto desde cualquier dispositivo en cualquier momento, ya que éstos están almacenados en la nube. Por ello, se puede trabajar sobre los mismos ficheros tanto en la versión de escritorio como en la online, instalando Matlab Drive Connector en el ordenador de sobremesa o portátil que se vaya a utilizar para este propósito.

Para poder otorgar la funcionalidad necesaria a la aplicación desarrollada para el presente trabajo, se ha requerido la instalación de los siguientes toolboxes o módulos:

- **Deep Learning Toolbox** (MathWorks). Utilizada para el diseño de redes LSTM para la clasificación de acciones.
- **MATLAB Support Package for Apple iOS Sensors** (MathWorks). Necesaria para poder capturar la información procedente de los sensores y proporcionada por la aplicación MATLAB Mobile.
- **Sensor Fusion and Tracking Toolbox** (MathWorks). Imprescindible para la simulación de sistemas que utilizan sensores.
- **Icons**. Usado para editar e incluir iconos en la interfaz gráfica de la aplicación.

También, conviene reseñar que posee una documentación online muy detallada, junto con la documentación propia, mencionada previamente y que en conjunto aportan numerosos ejemplos y vídeos tutoriales.

3.1.2 ThingSpeak

ThingSpeak también se ha utilizado bajo la licencia de estudiante. Se trata de una plataforma de análisis específicamente diseñada por MathWorks para IoT que permite almacenar datos en la nube y una perfecta comunicación y sincronización con las aplicaciones de Matlab.

Esta herramienta se compone de dos elementos principales, canales y aplicaciones. Los canales, que pueden ser privados o públicos, almacenan los datos recogidos de una aplicación para luego poder trabajar con ellos. Cada canal es identificado con un ID y dispone de ocho campos que pueden almacenar cualquier tipo de datos, tres campos de localización (latitud, longitud y elevación) y uno de estado de los datos. Asimismo, se le asocia una Write API Key y una Read API Key. Si el canal es privado, para poder acceder a los datos almacenados

en él es necesario indicar, aparte del ID del canal, la Read API Key para poder leerlos. Sin embargo para poder introducir datos en el canal, ya sea público o privado, hay que incluir la Write API Key.

Las aplicaciones complementan al proyecto que se esté desarrollando. La aplicación para gestionar y transformar los datos es Matlab Analysis, y para visualizarlos en diagramas Matlab Visualizations. Además, se puede vincular una cuenta de Twitter a una cuenta de ThingSpeak para así poder publicar mediante ThingTweet tweets en modo de alerta. Otras aplicaciones principales son TimeControl, React y ThingHTTP, cuyas finalidades principales, relacionadas con el presente trabajo, consisten en:

a) Programar la realización de una acción específica en un momento determinado.
b) Realizar una acción dadas unas condiciones previamente programadas en el canal.
c) Permitir la comunicación entre servicios web.

De las aplicaciones citadas, han sido necesarias para el desarrollo de este trabajo las siguientes: Matlab Analysis, ThingTweet y React.

3.2 Diseño de la aplicación

3.2.1 Arquitectura

La aplicación, como se puede observar en la figura 3.1, se divide principalmente en dos secciones: Matlab y ThingSpeak. En la sección anterior se han mencionado estos mismos elementos como herramientas. A continuación, se introducen desde el punto de vista de la funcionalidad que proporcionan. Las relaciones entre cada elemento están representadas por las líneas que se indican en dicha figura, constituyendo las conexiones entre ambas desde el punto de vista del diseño de la aplicación.

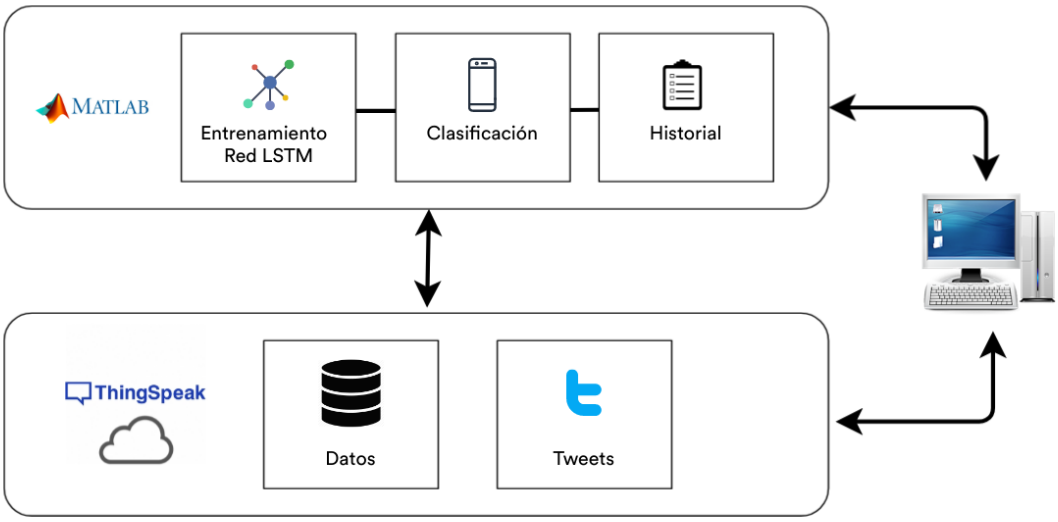


Figura 3.1 Arquitectura de la aplicación

- **Matlab**

Matlab abarca las funcionalidades básicas y principales de la aplicación, consistentes en el entrenamiento de la red LSTM y la clasificación de acciones una vez actualizados los pesos de las diferentes capas tras dicho entrenamiento. Una vez realizado el entrenamiento, se procede a activar los sensores en el dispositivo móvil y a capturar los datos de la aceleración para proceder a la clasificación de la acción ejecutada en ese momento y que constituyen las entradas a la red entrenada.

- **ThingSpeak**

ThingSpeak proporciona y habilita la posibilidad de gestión y almacenamiento de los datos en la nube. Cuando se completa la clasificación de acciones, se introducen los datos obtenidos de los sensores en el correspondiente canal de ThingSpeak así como la acción realizada, lo que sirve para poder acceder al historial de todas las acciones realizadas, además de posibilitar la publicación de los tweets correspondientes según la acción realizada. En este caso, se trata de una funcionalidad específica programada a través de la interfaz de aplicaciones que proporciona ThingSpeak.

3.2.2 Diseño

En la figura 3.2 se muestra la pantalla principal de la aplicación. En la parte izquierda se encuentran las ventanas de acceso a las distintas funcionalidades implementadas, a saber:

- Entrenamiento de la red.
- Preparación del entorno de operación, a través del cual se carga la red previamente entrenada, así como la identificación del dispositivo móvil que se utilizará para la captura de acciones a través de los sensores de movimiento.
- Clasificación de una acción en función de los valores de aceleración proporcionados por los sensores del dispositivo móvil.
- Acceso al historial de las actividades mediante los servicios de ThingSpeak.

En el panel de la parte derecha de la figura 3.2 se muestra un contendor relativa a una representación gráfica, en este momento vacía, de los datos de la aceleración y el nombre de la acción realizada, que se actualiza una vez terminado el proceso de clasificación.

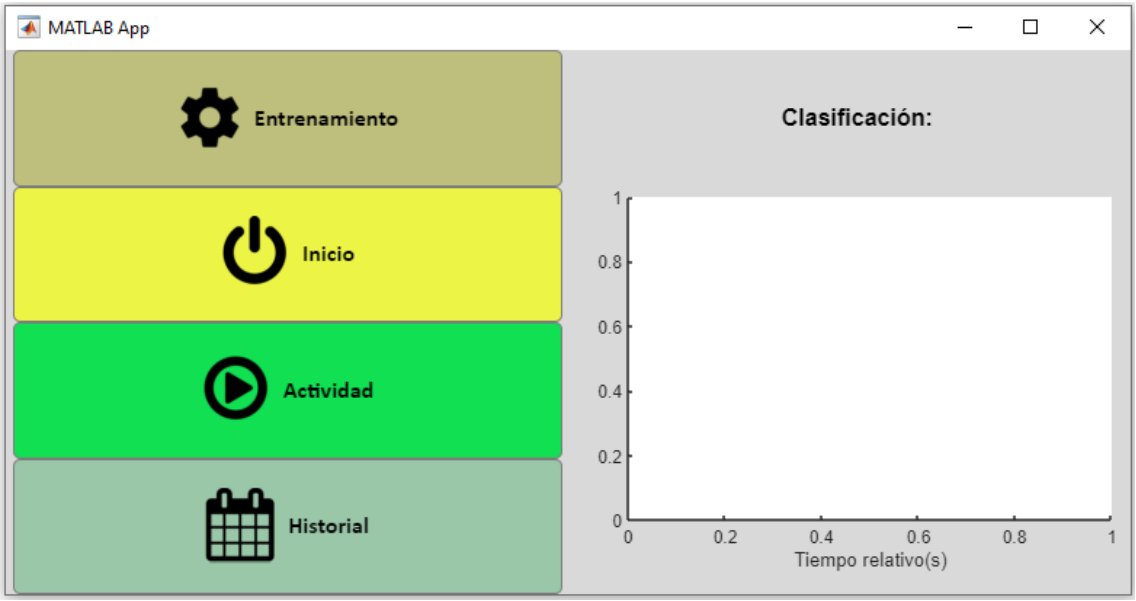


Figura 3.2 Vista principal de la aplicación

3.2.2.1 Funcionalidades

- **Entrenamiento**

Como se ha mencionado previamente, bajo el paradigma de Aprendizaje Profundo, se entrena una red de tipo LSTM con los datos de los sensores de movimiento, concretamente los correspondientes a la aceleración, obtenidos mediante un dispositivo móvil y utilizando las componentes del vector aceleración según sus correspondientes tres ejes X, Y, Z.

Para el entrenamiento se dispone de seis secuencias de datos de aceleración con el número de datos que se muestra en la tabla siguiente. Son datos disponibles en el repositorio de Matlab. En la figura 3.3 se muestra una representación de los datos de aceleración para la secuencia de entrenamiento número 3, con sus 56.416 datos. Todas las secuencias contienen datos correspondientes a las cinco acciones indicadas, que por otra parte constituyen las etiquetas utilizadas como referencia para el entrenamiento.

Secuencia	1	2	3	4	5	6
Número de datos	64.480	53.396	56.416	50.688	51.888	54.256

Tabla 3.1 Secuencias y número de datos de aceleración

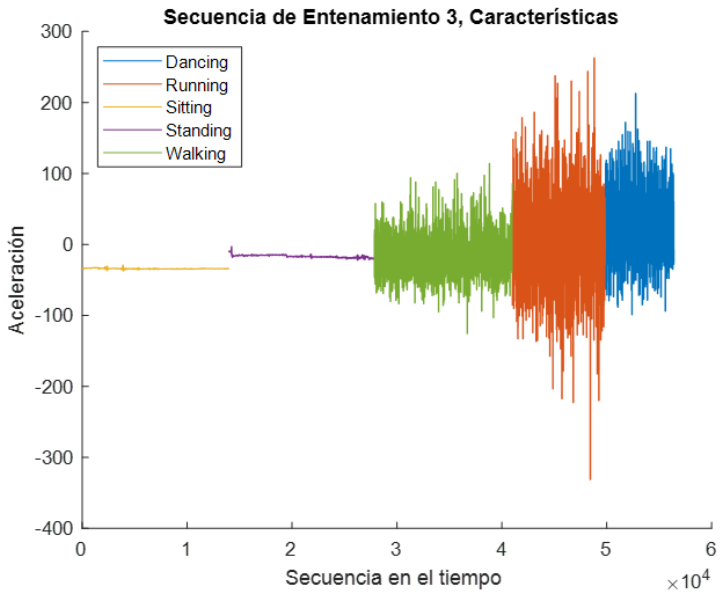


Figura 3.3 Representación de datos de entrenamiento para la secuencia 3

Cuando los datos de reconocimiento de la actividad humana están cargados, se define la arquitectura de la red LSTM indicando como entrada secuencias de tamaño 3, especificando una capa LSTM con 200 unidades ocultas y generando una secuencia completa tal y como se

muestra en la figura 3.4, S2S (2022). Donde la secuencia de entrada (*sequenceinput*) constituye el nodo de entrada, que conecta con la mencionada estructura LSTM y a partir de aquí las salidas se proyectan sobre una capa totalmente conectada (*fc*, *fully connected*), que a su vez se enlaza con la capa *softmax* para determinar las probabilidades de pertenencia de cada dato de aceleración a una de las clases previstas (*Dancing, Running, Sitting, Standing, Walking*), de forma que en la capa final *classoutput* se determina la acción correspondiente en función de la máxima probabilidad obtenida en la mencionada capa *softmax*.

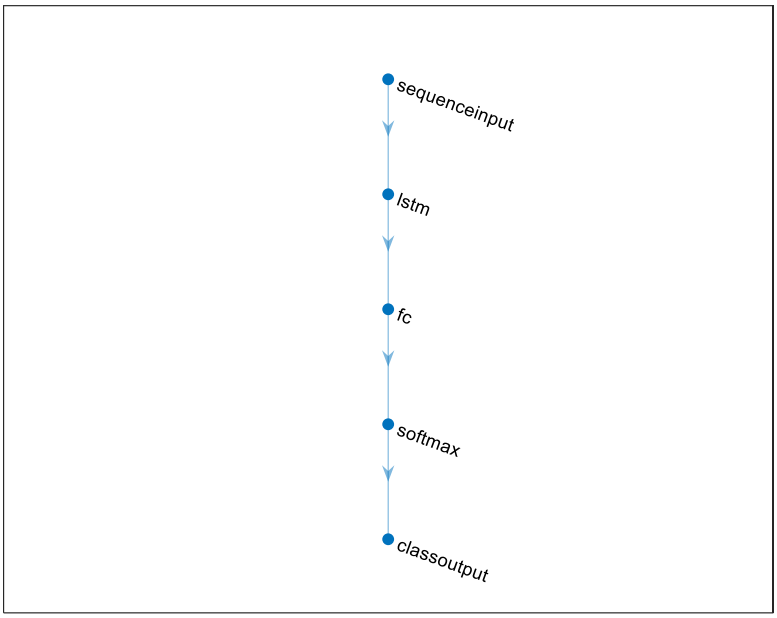


Figura 3.4 Modelo de la red con la estructura LSTM insertada

Continuando con la figura 3.5, se muestra el panel correspondiente a lo que se denomina *Solver*, la Tasa o razón inicial de aprendizaje y el Número máximo de épocas del entrenamiento por defecto, si bien tal y como se ha definido la aplicación estos valores son modificables.

El *Solver*, que se refiere al algoritmo de optimización utilizado, da la opción de elegir uno de los siguientes valores (Pajares y col., 2021):

- sgdm (stochastic gradient descent with momentum) – utiliza el optimizador SGDM.
- adam – utiliza el optimizador Adam.
- rmsprop – utiliza el optimizador RMSProp.

El número máximo de épocas es el número de veces que el algoritmo procesa todo el conjunto de entrenamiento.

Respecto a la razón de aprendizaje inicial, si es muy baja el entrenamiento se extenderá con un mayor número de iteraciones, y si es muy alta el resultado puede llegar a ser

subóptimo o divergir. Es necesario, por tanto, buscar un equilibrio dentro del rango de valores permitido, esto es [0,1], sin que exista un criterio claro de cuál debe ser el mejor valor, TrainingOptions (2022).

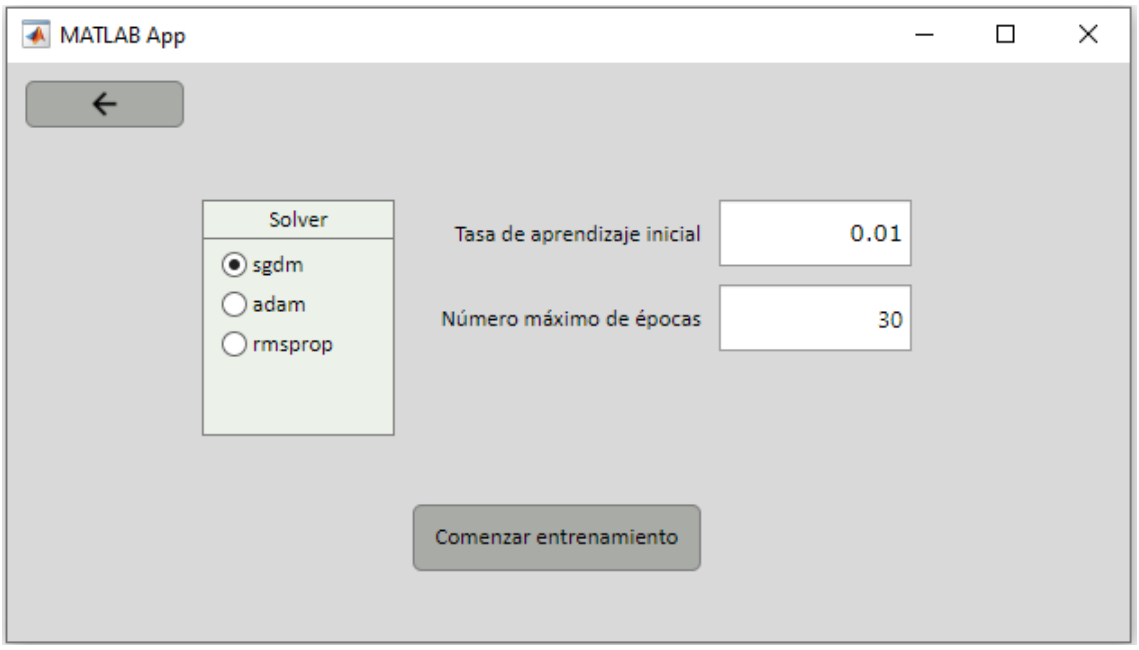


Figura 3.5 Opciones de entrenamiento de la red

- **Inicio**

Con anterioridad a la clasificación, para recibir los datos que envían los sensores de un dispositivo móvil es necesario identificar dicho dispositivo. La cuenta en la que se inicie sesión de Matlab Desktop o Matlab Online debe ser la misma que la de Matlab Mobile.

- **Actividad**

Utilizando la red LSTM ya entrenada y activando los sensores en el móvil, se procede a la clasificación de la actividad desde la versión de escritorio u online de Matlab, teniendo en cuenta para ello sólo los datos de aceleración en sus tres componentes X, Y y Z, coincidiendo con los correspondientes valores de entrenamiento.

- **Historial**

Se puede llevar a cabo un seguimiento de las acciones realizadas con una tabla que señala la fecha y la actividad, como se observa en la figura 3.6. Las fechas estarán ordenadas de mayor a menor, teniendo así la más reciente en la primera fila. Esta funcionalidad corresponde al diseño de ThingSpeak.

Con tal propósito, el canal creado en ThingSpeak consta de cuatro campos, en los que se introducen datos cuando se obtiene el tipo de actividad a través del proceso de clasificación en Matlab con la red LSTM. Estos campos se corresponden exactamente con los tres ejes mencionados de la aceleración y la clasificación de la acción correspondiente, como se muestra en la figura 3.6.

Cuando se accede al historial, se obtienen los datos previamente introducidos en el canal de ThingSpeak, figura 3.7.

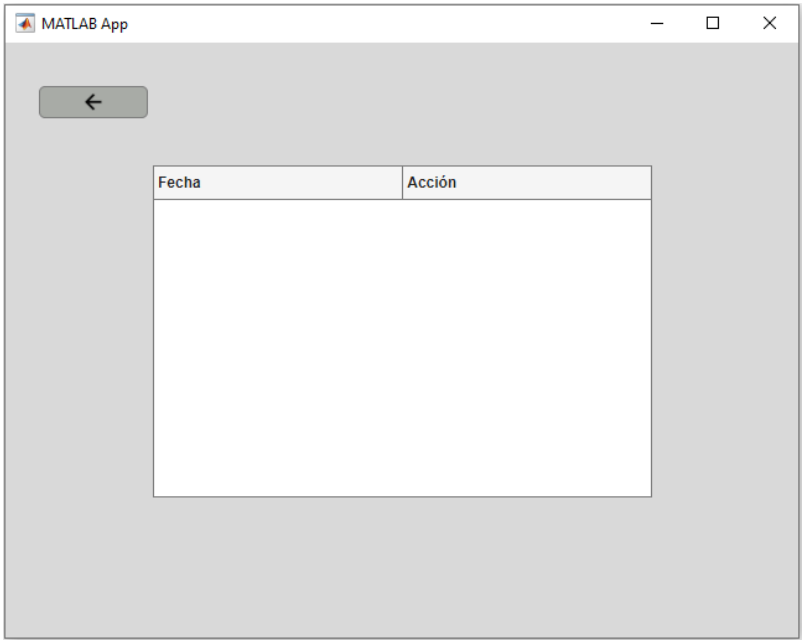


Figura 3.6 Vista del historial de acciones

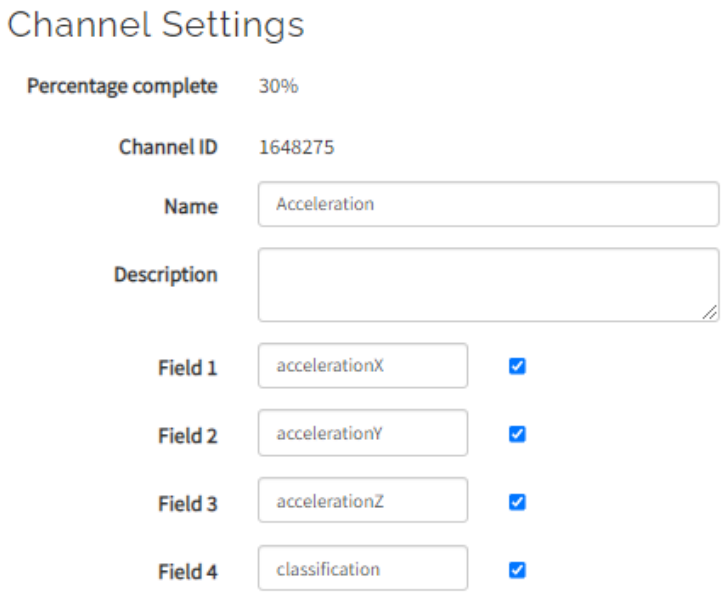


Figura 3.7 Campos del canal de ThingSpeak

- **Twitter**

Cada vez que se registra un movimiento en ThingSpeak, se publica un tweet por medio de las aplicaciones ThingTweet y React. Los tweets se indican por medio de la última aplicación mencionada, de forma que cada acción tiene el suyo propio, excepto cuando se detecta que el usuario está sentado (*Sitting*), ya que en este caso se publica un tweet cuando se obtiene esta actividad dos o más veces seguidas, que corresponde a la reacción "Sin movimiento" de la figura 3.8. Cada una de las acciones se especifica tal y como se muestra en el ejemplo de la figura 3.9.

Name	Created	Last Ran
Bailar View Edit	2022-02-12	2022-04-25 5:01 pm
Correr View Edit	2022-02-23	2022-04-25 4:44 pm
Andar View Edit	2022-02-23	2022-04-25 4:54 pm
Parado View Edit	2022-02-23	2022-04-25 5:08 pm
Sin movimiento View Edit	2022-02-24	2022-04-25 4:25 pm

Figura 3.8 Reacciones de la aplicación React de ThingSpeak vinculadas a Twitter

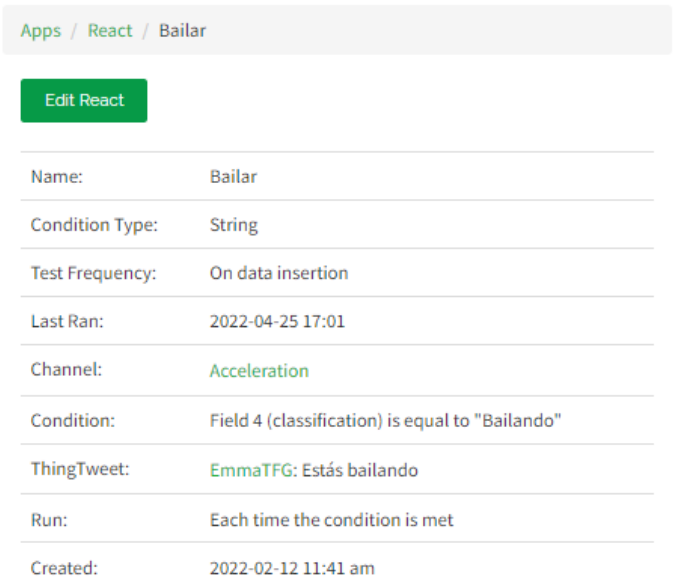


Figura 3.9 Reacción del movimiento "Bailar" en React

En ThingTweet se encuentra la cuenta de Twitter vinculada con el usuario @EmmaTFG y las reacciones de ésta, figura 3.10.

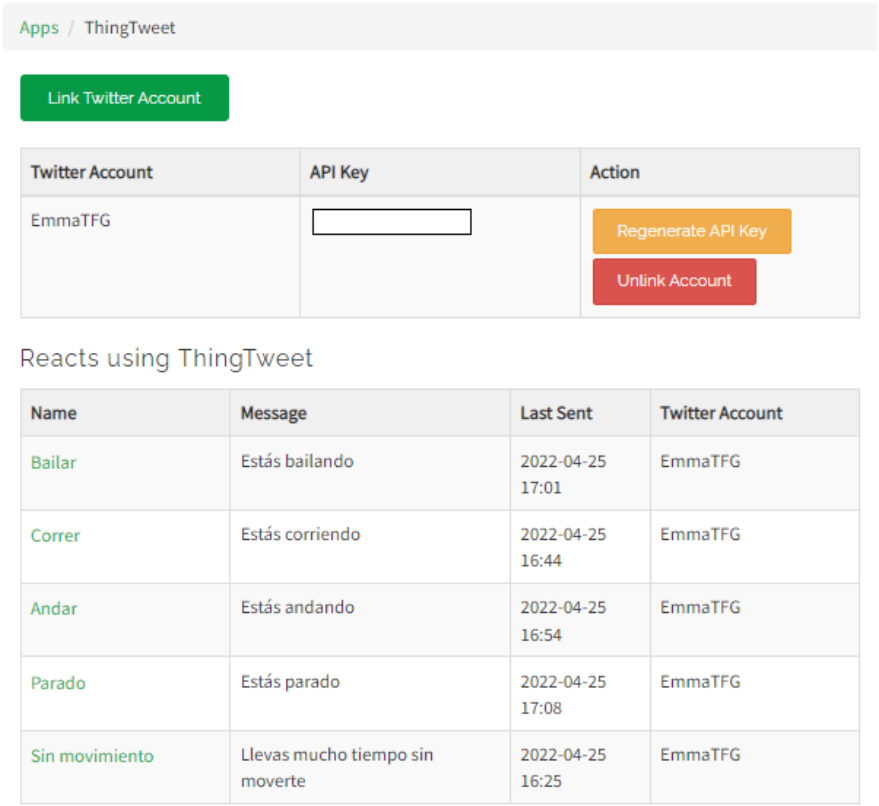


Figura 3.10 ThingTweet

Capítulo 4. Resultados

En este capítulo se muestran los resultados de la aplicación según las funcionalidades de la aplicación descritas anteriormente.

4.1 Entrenamiento

Por defecto se dispone de una red previamente entrenada con la función de optimización *adam*, utilizando 60 épocas como valor máximo y 0.001 como razón de aprendizaje inicial.

Si se quisiera cambiar cualquiera de estos valores, se introduce en cada cuadro de texto de la ventana correspondiente el valor que se desee y se pulsa sobre "Comenzar entrenamiento". En total aparecerán tres gráficas, una al iniciar este proceso y otras dos al finalizarlo, figura 4.1. En la figura 4.1(*a*) se muestra una secuencia de datos previamente capturados con el dispositivo móvil, correspondientes a valores de aceleración con los que se entrena el modelo de red. Se identifican por su color los datos de aceleración correspondientes a cada una de las cinco acciones que reconoce la red LSTM (Dancing, Running, Sitting, Standing, Walking), capturados a lo largo del tiempo. Por otro lado, se dispone también de una serie de datos de test, igualmente previamente capturados y almacenados correspondientes a valores de aceleración para el mismo tipo de acciones previstas. En esta gráfica aparecen representadas de forma superpuesta tres tipos de datos de test (Características 1, 2 y 3) a lo largo del tiempo. Utilizando estos datos de test se procede a la validación de los mismos clasificando las acciones correspondientes con el modelo entrenado, de esta forma se realiza una predicción de los resultados en cuanto a las acciones identificadas y a la vez se comparan con las acciones verdaderas, que aparecen recogidas también en los datos de test. Es decir, los datos de test contienen la acción que corresponde a cada terna de datos de aceleración. Esto permite comparar los resultados de la clasificación del modelo LSTM con los de test, en lo que se conoce como validación del modelo. De hecho, en el gráfico de la figura 4.1(*c*) se representan solapados tanto los datos predichos o clasificados con el modelo, identificados como "Predicción", como los verdaderos, que se identifican como "Datos test". La forma gráfica de determinar el grado de acierto consiste en analizar las discrepancias de color según las gráficas de predicción y datos de test, observándose una muy alta coincidencia, ya que las predicciones (gráfica azul) ocultan los valores del test (gráfica naranja).

Si antes de comenzar el entrenamiento no se dispone del archivo de red previamente almacenado, se creará uno nuevo tras la finalización del entrenamiento. En caso contrario, se sobrescribirá el existente, actualizándose el modelo LSTM de red entrenado.

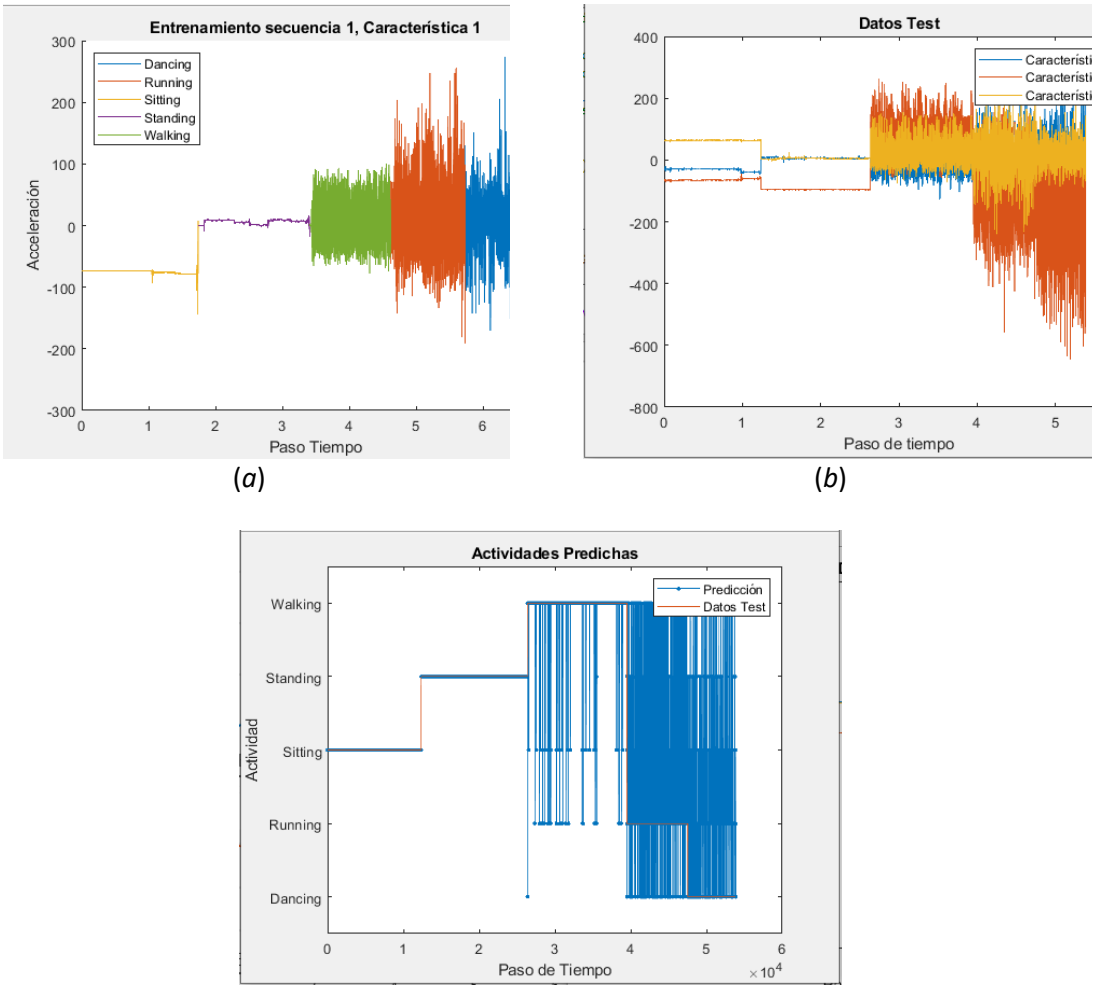


Figura 4.1. (a) Datos de entrenamiento del modelo LSTM; (b) Datos de test; (c) Actividades predichas

Al empezar el entrenamiento también emergerá una ventana indicando el progreso del éste. Por ejemplo, poniendo como épocas máximas 10, como solver sgdm y la tasa de aprendizaje inicial al valor 0.01, aparecería la ventana mostrada en la figura 4.2. En la parte izquierda se muestran dos gráficas, una correspondiente a la evolución de la precisión y otra a la función de pérdida, y en la parte derecha la información relativa al entrenamiento, así como la leyenda de las líneas de cada gráfica.

Como se ha indicado previamente, las gráficas sirven para determinar el progreso de la precisión y pérdida de la red. Cada una de ellas muestran tres tipos de datos, aunque en esta

aplicación sólo se utilizan los dos primeros: el entrenamiento, el entrenamiento suavizado y la validación.

El entrenamiento señala la precisión o pérdida de la clasificación de cada mini-lote y al suavizado se le aplica un algoritmo para identificar la detección de tendencias.

Se puede parar este proceso pulsando sobre el botón que se ubica al lado de la barra que indica el número de iteración en la que se encuentra actualmente. Haciendo esto, el archivo con el modelo de red previamente entrenado no se verá modificado y por tanto, no se actualiza para los nuevos datos de entrenamiento.

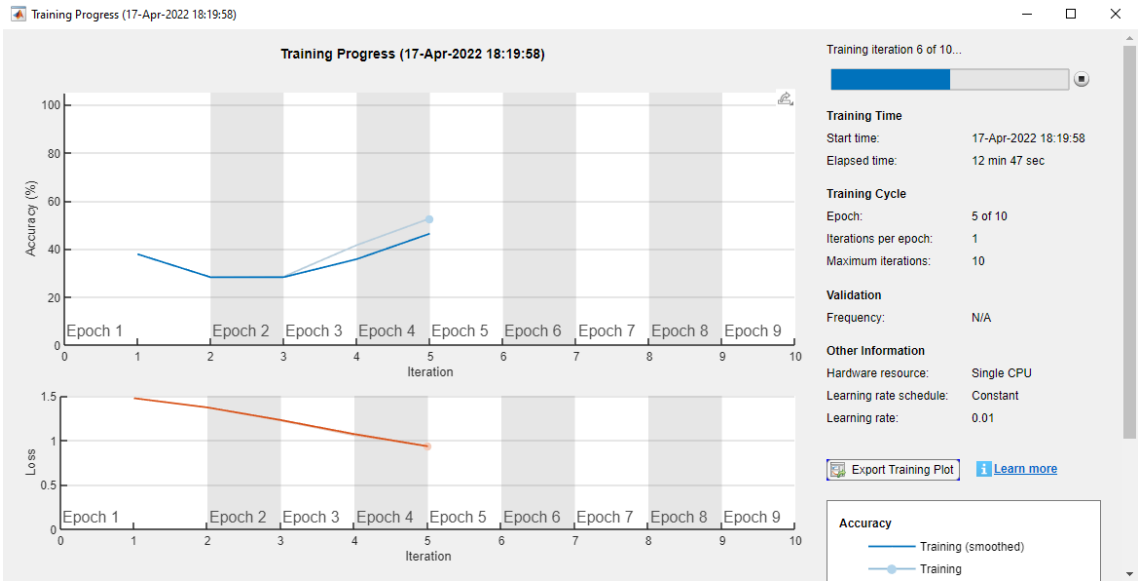


Figura 4.2 Progreso del entrenamiento

En la vista correspondiente a la configuración del entrenamiento, en el caso de introducir datos no válidos como puede ser una letra o un valor negativo en cualquiera de las dos casillas, se indicará con un mensaje de advertencia como el de la figura 4.3, y el recuadro volverá a tener su valor por defecto, en este caso 30.

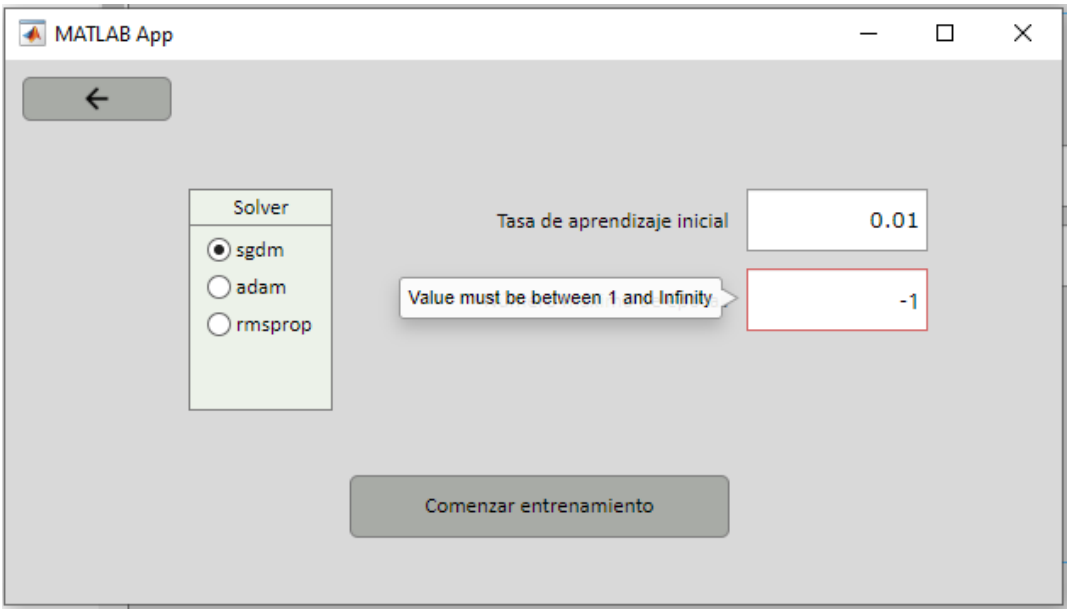


Figura 4.3 Mensaje de error al introducir un número negativo como número de épocas

4.2 Clasificación

Como se ha explicado previamente, para realizar la clasificación de una acción es necesario activar los sensores del dispositivo móvil que se vaya a utilizar para la captura de los datos de aceleración e iniciar la aplicación. Por lo tanto, se gestionan distintos tipos de errores:

- Al pulsar sobre el botón Inicio, saltará un error si no existe un archivo de red previamente entrenada o si no se detecta ningún dispositivo móvil.
- Al pulsar sobre Actividad se distinguen tres errores:
  - Si no se ha iniciado.
  - Si se ha iniciado pero no se detecta el móvil.
  - Si se ha iniciado y se detecta el móvil pero no están activados los sensores.

En la figura 4.4 se puede ver el caso en el que no se han activado todos los sensores.

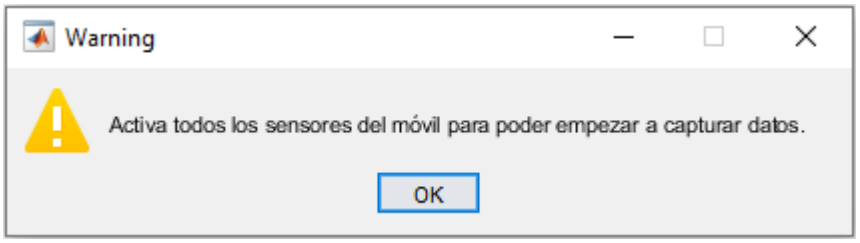


Figura 4.4 Mensaje que indica que hay que activar todos los sensores para poder iniciar la captura de datos

Un ejemplo de clasificación de movimiento sería el "sentado" (*Sitting*) como el mostrado en la figura 4.5, en la que se puede observar el nombre del movimiento clasificado con los datos correspondientes de aceleración proporcionados por el sensor y según una de las componentes de aceleración, en este caso Aceleración en el eje X. Al tratarse de una acción de escaso o nulo movimiento, las variaciones de aceleración son pequeñas. Para otros movimientos estos valores poseen componentes mayores.

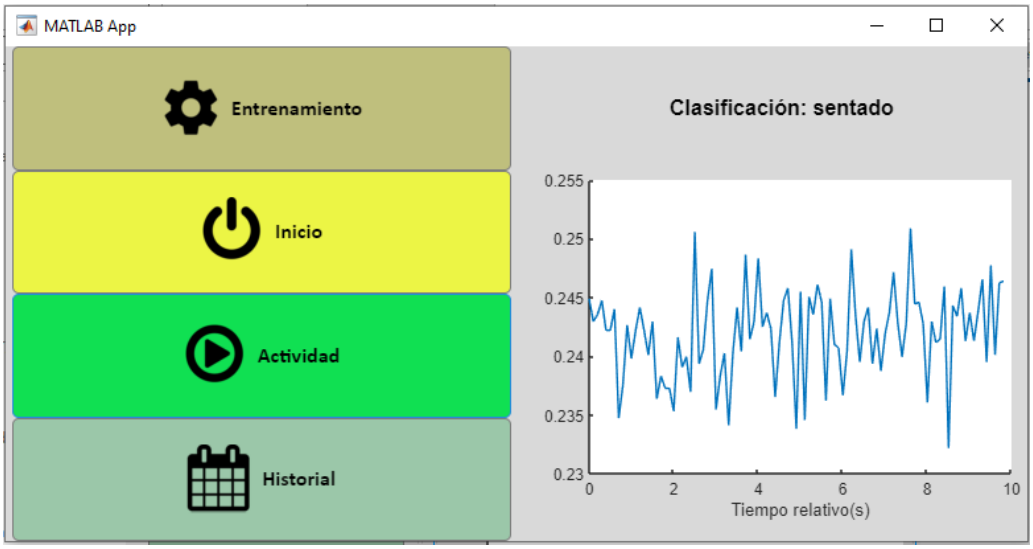


Figura 4.5 Clasificación de la acción 'Sitting'

4.3 Twitter

Como ya se vio en el capítulo anterior, cada vez que se clasifica un movimiento se inserta la acción correspondiente en el canal asociado de ThingSpeak.

La figura 4.6 muestra los datos del canal en las tres gráficas que representan los tres ejes de la aceleración (X, Y, Z).

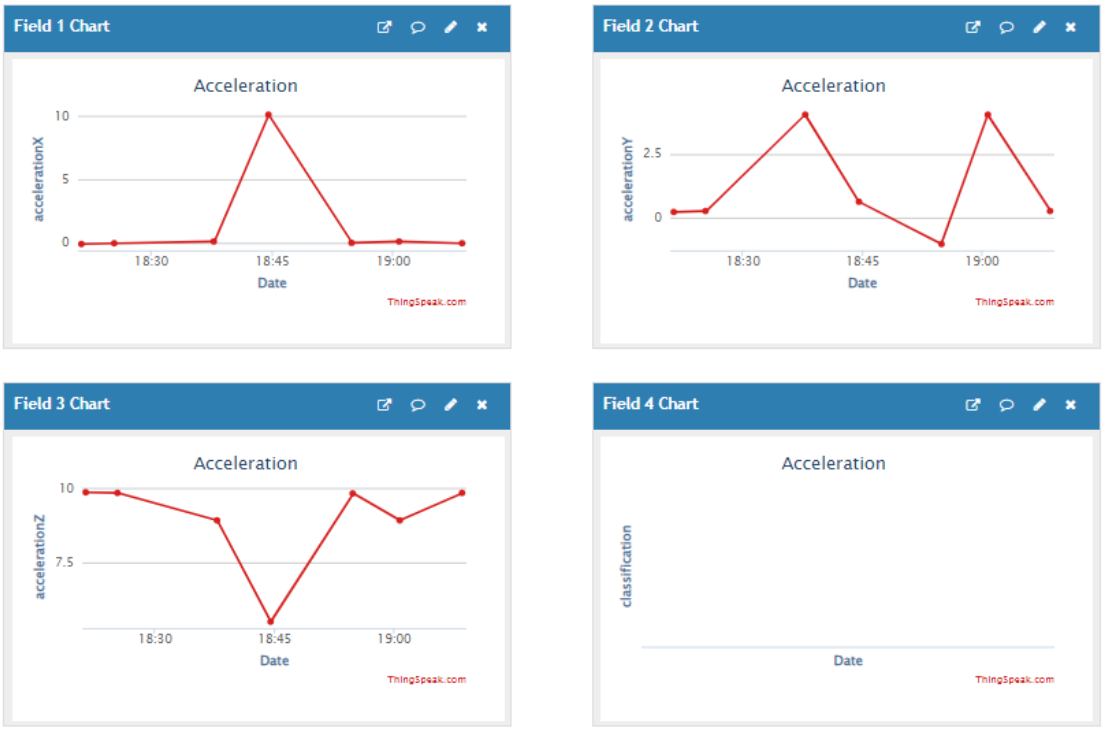


Figura 4.6 Gráficas de cada eje de aceleración

React y ThingTweet se encargan de twittear la acción realizada, en la figura 4.7 se muestra un ejemplo de la cuenta de twitter asociada a ThingSpeak, de forma que según la acción realizada así es el mensaje correspondiente, tal y como se observa en dicha figura.

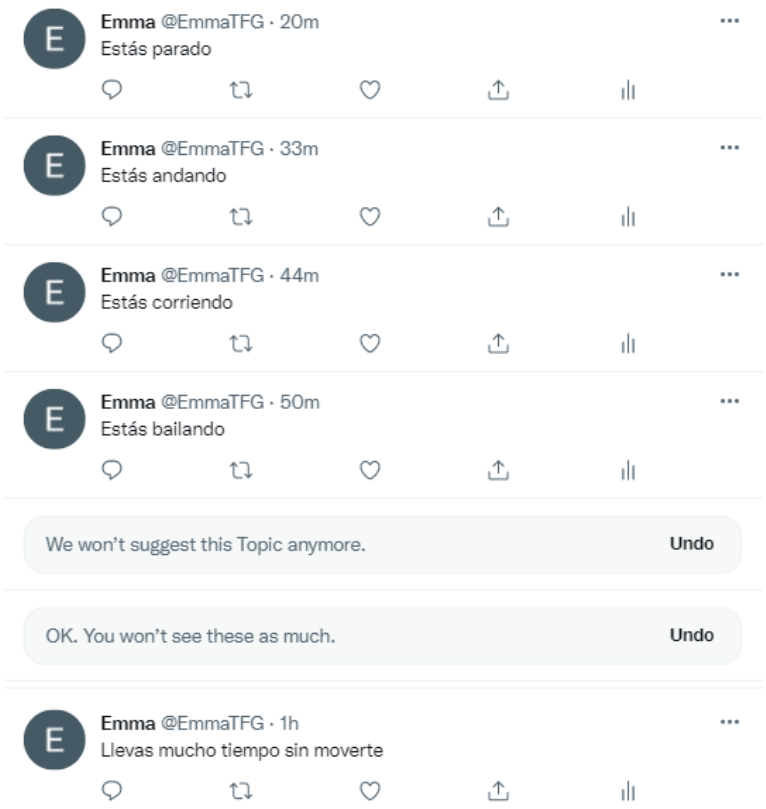


Figura 4.7 Tweets de la cuenta de twitter asociada a ThingSpeak

4.4 Historial

Una vez realizadas una o más acciones y clasificadas utilizando el modelo de red entrenado, estas se pueden ver en el historial de acciones, figura 4.8.

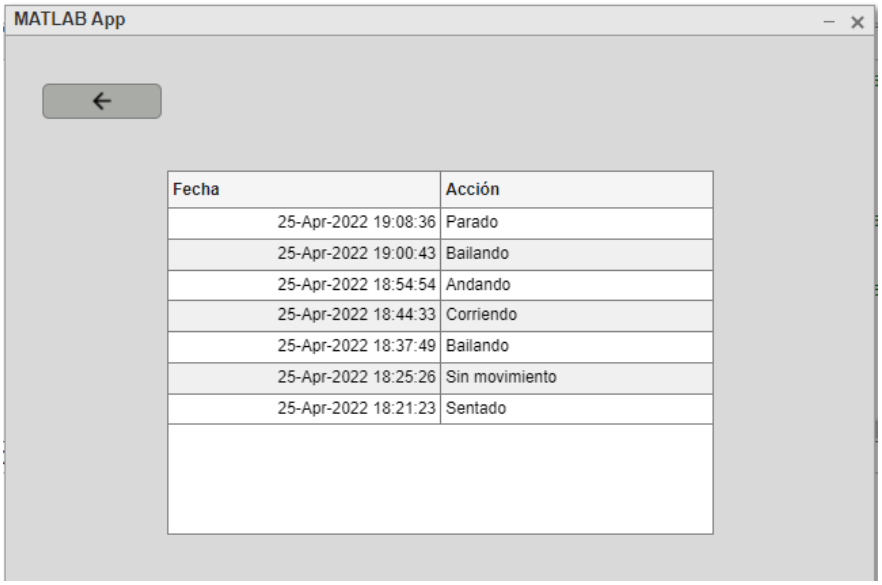


Figura 4.8 Historial de acciones realizadas

Capítulo 5. Conclusiones y trabajo futuro

5.1. Conclusiones

Gracias a la utilización de las herramientas que proporciona Matlab y ThingSpeak ha sido posible desarrollar la aplicación que se presenta, que consiste en la clasificación de acciones humanas según el movimiento realizado.

Los datos de movimiento son captados por los sensores de un dispositivo móvil, concretamente son valores de aceleración en los tres ejes X, Y, Z.

El diseño de la aplicación consiste en una interfaz de usuario, que da acceso a los procesos implementados bajo su cobertura. Uno de tales procesos es el entrenamiento de una red del tipo LSTM, dentro del paradigma de Aprendizaje Profundo, cuya arquitectura consiste en una serie de celdas secuenciales en el tiempo, teniendo como entrada los mencionados datos de aceleración. Un segundo proceso es la clasificación de nuevas acciones, utilizando también datos de aceleración, obviamente una vez entrenada la red. Ambas funcionalidades se encuentran desarrolladas en Matlab.

Una tercera funcionalidad asociada consiste en la conexión a un servidor en la nube para el registro y almacenamiento de datos, así como el historial de acciones realizadas. Esto se realiza mediante los servicios ofrecidos por la plataforma ThingSpeak, especialmente diseñada para trabajar bajo el paradigma de IoT. Aprovechando los recursos y servicios ofrecidos por ThingSpeak se ha programado un evento que conecta con la red social Twitter y que permite recibir tweets en una cuenta a dicha red conteniendo información de las acciones realizadas. Todo esto con la consiguiente cobertura de internet.

Las distintas pruebas realizadas han permitido comprobar el correcto funcionamiento de la aplicación desarrollada para los distintos tipos de movimiento previstos con los que la red ha sido entrenada.

5.2 Trabajo futuro

De cara al futuro, la aplicación podría mejorarse realizando los siguientes cambios y ampliaciones detectadas durante el desarrollo:

- Añadir más avisos a través de tweets, por ejemplo, si no se ha detectado movimiento en un día entero, como alerta ante situaciones no deseadas.
- Ofrecer la posibilidad de entrenar los modelos LSTM con distintos tipos de acciones, no necesariamente las establecidas por defecto. De esta forma, para la clasificación se podrán utilizar distintos tipos de red entrenada a elección del usuario.
- Añadir, mediante la interfaz de usuario, la posibilidad de introducir todos los parámetros configurables para el entrenamiento, citando expresamente el umbral de gradiente o la frecuencia de validación de la red, cuyo objetivo es poder obtener una red entrenada más personalizada.
- Añadir más tipos de movimientos, que se añadan a la base de movimientos preestablecidos.
- Añadir una nueva funcionalidad para para resetear el historial de movimientos, borrando los campos del canal correspondiente en ThingSpeak.
- Añadir estadísticas de los movimientos realizados, cada cierto periodo de tiempo, por ejemplo diario, semanal o mensual, incluyendo gráficas y porcentajes actualizados.

Chapter 5. Conclusions and future work

5.1 Conclusions

Thanks to the use of the tools provided by Matlab and ThingSpeak, it has been possible to develop the application presented, which consists of the classification of human actions according to the movement performed.

The motion data are captured by the sensors of a mobile device, specifically acceleration values in the three axes X, Y, Z.

The application design consists of a user interface, which gives access to the processes implemented under its coverage. One such process is the training of a network of the LSTM type, within the Deep Learning paradigm, whose architecture consists of a series of sequential cells in time, having as input the aforementioned acceleration data. A second process is the classification of new actions, also using acceleration data, obviously once the network has been trained. Both functionalities are developed in Matlab.

A third associated functionality consists of the connection to a cloud server for data logging and storage, as well as the history of actions performed. This is done through the services offered by the ThingSpeak platform, specially designed to work under the IoT paradigm. Taking advantage of the resources and services offered by ThingSpeak, an event has been programmed that connects with the social network Twitter and allows receiving tweets in an account to that network containing information of the actions performed. All this with the consequent internet coverage.

The tests carried out have allowed to verify the correct functioning of the application developed for the different types of movements foreseen with which the network has been trained.

5.2 Future work

Looking ahead, the app could be improved by making the following changes and extensions discovered during development:

- Add more warnings through tweets, for example, if no movement has been detected for a whole day, as an alert to unwanted situations.

- Offer the possibility of training LSTM models with different types of actions, not necessarily those established by default. In this way, for the classification, different types of trained network can be used at the user's choice.
- Add, through the user interface, the possibility of introducing all the configurable parameters for training, expressly citing the gradient threshold or the network validation frequency, whose objective is to be able to obtain a more personalized trained network.
- Add more types of movements, which are added to the base of pre-established movements.
- Add a new functionality to reset the history of movements, deleting the fields of the corresponding channel in ThingSpeak.
- Add statistics of the movements made, every certain period of time, for example daily, weekly or monthly, including graphs and updated percentages.

Bibliografía

1. Bishop, C.M. (2006). Pattern Recognition and Machine Learning, Springer, NY, USA.
2. Gers, F.A, Schraudolph, N.N., Schmidhuber, J. (2002). Learning precise timing with LSTM recurrent networks. Journal of Machine Learning Research, 3,115–143.
3. Goodfellow, I., Bengio, Y., Courville, A. (2016). Deep learning. MIT Press. Disponible on-line: https://www.deeplearningbook.org/ (Accedido Marzo 2022).
4. Graves, A. (2013). Generating sequences with recurrent neural networks, Computer Science. arXiv:1308.0850.
5. Hochreiter, S., Schmidhuber, J. (1997). Long short-term memory. Neural Computation, 9(8), 1735–1780.
6. Olah, C. (2015). Understanding LSTM Networks. Disponible on-line: http://colah.github.io/posts/2015-08-Understanding-LSTMs/ (Accedido Marzo 2022).
7. Pajares, G., Herrera, P.J., Besada, E. (2021). Aprendizaje Profundo. RC-Libros, Madrid.
8. Rego-Drumond, R., Dorta-Marques, B., Nader-Vasconcelos, C. Clua, E. (2018). PEEK - An LSTM Recurrent Network for Motion Classification from Sparse Data. In Proc. 13th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 1: GRAPP, pp. 215-222.
9. Rumelhart, D.E., Hinton, G.E., Williams, R.J. (1986). Learning representations by back-propagating errors. Nature. 323 (6088), 533–536.
10. Ruder, S. (2017). An overview of gradient descent optimization algorithms. arXiv:1609.04747v2 [cs.LG].
11. S2S (2022) Sequence-to-Sequence Classification Using Deep Learning. Disponible on-line: https://es.mathworks.com/help/deeplearning/ug/sequence-to-sequence-classification-using-deep-learning.html (Accedido Abril 2022).
12. AI (2022). Artificial intelligence. The Alan Turing Institute. Disponible on-line: https://www.turing.ac.uk/research/research-programmes/artificial-intelligence-ai (Accedido Abril 2022).
13. Casado-Fernández, M.C. (2022). Manual Básico de Matlab. Servicios Informáticos U.C.M. Apoyo a Investigación y Docencia. Disponible on-line: https://webs.ucm.es/centros/cont/descargas/documento11541.pdf (Accedido Abril 2022).

14. Matlab (2022). Descripción del producto MATLAB. Disponible on-line: https://es.mathworks.com/help/matlab/learn_matlab/product-description.html (Accedido Abril 2022).
15. TrainingOptions (2022). Options for training deep learning neural network. Disponible on-line: https://es.mathworks.com/help/deeplearning/ref/trainingoptions.html (Accedido Abril 2022).

ANEXO: Manual de usuario

A continuación se describen los pasos necesarios y las instrucciones correspondientes para la ejecución de la aplicación, cuyo código se encuentra en el siguiente enlace:

https://github.com/emmatalavera/TFG.git

Pasos previos

Para poder ejecutar la aplicación, es necesario seguir previamente estos pasos:

1. Descargar Matlab Desktop.
2. Descargar Matlab Drive.
3. Instalar en el móvil Matlab Mobile.
4. Instalar en Matlab Desktop los siguientes toolboxes: Deep Learning Toolbox, MATLAB Support Package for Apple iOS Sensors o MATLAB Support Package for Apple Android Sensors, Sensor Fusion and Tracking Toolbox.

Instrucciones

Entrenamiento

Como ya se ha explicado anteriormente, se dispone de un archivo de red entrenada por defecto llamado "RedLSTM" con la función de optimización *adam*, utilizando 60 épocas como valor máximo y 0.001 como razón de aprendizaje inicial.

Si se quiere modificar algún campo, se introducen los nuevos valores y se debe pulsar sobre "Comenzar entrenamiento". Esto sobrescribirá el archivo existente.

Si se desea, se puede parar este proceso para que el archivo no se vea modificado, pulsando sobre el botón de stop situado en la parte de arriba a la derecha.

Clasificación

Para llevar a cabo la clasificación, se debe abrir la aplicación Matlab en el dispositivo móvil que se vaya a usar y activar todos los sensores. Una vez hecho esto, se pulsará sobre el botón "Inicio" de la página principal y por último "Clasificación".

La clasificación aparecerá en la parte derecha de la vista, junto con el diagrama correspondiente a la aceleración. Además, se publicará un tweet con la acción realizada en la cuenta que está asociada a ThingSpeak (@EmmaTFG).

Historial

Para ver el historial de movimientos, pulsar sobre “Historial” de la página principal.