UNIVERSIDAD NACIONAL DE EDUCACIÓN A DISTANCIA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

Trabajo Fin de Máster en
Ingeniería y Ciencia de Datos

Clasificación de operaciones de embalaje por parte de operarios para el OpenPack Challenge 2022

Enrique Manuel Pérez Roa

Dirigido por: Olga Santos Martín

Curso: 2022-2023: 1ª Convocatoria

Agradecimientos

A todos los que me han ayudado en este proceso.

Resumen

El Reconocimiento de la Actividad Humana (HAR, por sus siglas en inglés) es una disciplina dentro del aprendizaje automático que busca reconocer y clasificar actividades desempañadas por seres humanos utilizando para ello distintas fuentes de datos. El objetivo del Trabajo de Fin de Máster es desarrollar un modelo capaz de realizar Reconocimiento de la Actividad Humana en el marco del *Open Pack Challenge* 2022. *OpenPack Challenge* 2022 es una competición de reconocimiento de diez tipos distintos de actividades que se dan durante el proceso de embalaje por parte de operarios en un almacén. Los participantes que participan en la competición tratan de presentar el sistema que mejor realice la clasificación de cada una de estas actividades, que son una serie de movimientos que realizan los trabajadores en el desarrollo de su trabajo. La información acerca de cada uno de estos movimientos viene proporcionada por un conjunto de sensores (aceleración, temperatura, etc...) que los propios operarios portaban. Al final del proyecto, el resultado será un modelo que, tomando como entrada estos datos ordenados temporalmente, sea capaz de clasificar con un alto grado de precisión la clase (actividad) a la que pertenecen. A fecha de la realización del trabajo la competición ya ha finalizado, por lo que la realización del TFM servirá para comparar la solución desarrollada con las que alcanzaron mejor puntuación, y no para participar directamente.

Palabras clave: Reconocimiento de la Actividad Humana, Aprendizaje profundo, Fusión de datos, Desafío

Abstract

Human Activity Recognition (HAR) is a discipline within machine learning that aims to recognize and classify activities performed by humans using various sources of data. The goal of the Master's Thesis is to develop a model capable of performing Human Activity Recognition within the framework of the Open Pack Challenge 2022. The Open Pack Challenge 2022 is a competition to recognize ten different types of activities that occur during the packaging process by warehouse workers. Participants in the competition aim to present the system that best classifies each of these activities, which are a series of movements carried out by the workers in the course of their work. Information about each of these movements is provided by a series of sensors (acceleration, temperature, etc.) that the workers themselves wore. At the end of the project, the result will be a model that, taking these temporally ordered data as input, is capable of classifying with a high degree of accuracy the class (activity) to which they belong. As of the completion of the work, the competition has already concluded, so the completion of the Master's Thesis will serve to compare the developed solution with those that achieved the highest scores, rather than participating directly.

Keywords: Human Activity Recognition, Deep Learning, Data fusion, Challenge

Índice de figuras

Índice de tablas

Índice general

Capítulo 1

Introducción

El objetivo del presente trabajo consiste en desarrollar una propuesta para el *Open Pack Challenge 2022*. que tuvo lugar durante la *IEEE International Conference on Pervasive Computing and Communications*. Los llamados *challenge* consisten en competiciones en las que se plantea un problema acerca de algún tema determinado, y los participantes tratan de encontrar la mejor solución posible. En el caso del *machine learning*, normalmente se suele proporcionar un conjunto de datos sobre el cuál los participantes tienen que desarrollar un modelo. Posteriormente, cada uno de los modelos desarrollados se evalúa con respecto a un conjunto de datos exclusivo de los organizadores, y las soluciones se puntúan en base a una métrica específica (precisión, cobertura, F1...). Puede existir alguna clase de recompensa para las mejores soluciones, en este caso eran, por un lado, una recompensa económica para cada una de las soluciones que quedaran entre los tres primeros puestos, y por otro el viaje al ciclo de conferencias para un miembro de cada uno de los equipos que finalizaran entre las cinco primeras posiciones. El desafío concluyó el 15 de enero de 2023, por lo que el objetivo del trabajo no es participar directamente, sino desarrollar una solución al problema planteado utilizando el conjunto de datos y las indicaciones del desafío, estudiar las opciones posibles para completarlo, y, dado que ya ha finalizado y están accesibles públicamente algunas de las soluciones planteadas (incluyendo la ganadora), compararlas ente sí. Toda la información puede encontrarse en [1].

El objetivo del desafío consiste en la clasificación de una serie de tareas que se realizan por parte de operarios en factorías o centros logísticos, tales como leer las etiquetas de una caja, ensamblar, empaquetar, etc... Cada una de estas actividades distintas están compuestas de múltiples acciones individuales, que la clasificación deberá tener en cuenta para llegar al valor de clase final, con una resolución de hasta un segundo, es decir, que para cada intervalo de un segundo en cada operación, el modelo debe clasificar correctamente a qué clase de operación pertenece.

El desarrollo del *Open Pack Challenge 2022* se enmarca dentro del campo más general del Reconocimiento de Actividades Humanas. De acuerdo con [17], el reconocimiento de actividades humanas consiste en la clasificación de actividades por medio de la Inteligencia Artificial, a partir de datos que provengan de una serie de fuentes llamadas sensores. Estos sensores pueden ser de tipos muy distintos (vídeo, detección de movimiento, sensores de temperatura, inerciales, etc...). Para este proyecto en particular, los datos crudos se han obtenido utilizando *Kinect* y *Intel RealSense DepthCamera D435i* (para la profundidad de imágen), *ATR TSND 151* como portable para aceleración y giroscopio, en las muñecas y en los brazos, y *E4 Wristband* para la

[1] https://open-pack.github.io/challenge2022

temperatura, también en las muñecas y brazos.

Debido a que los distintos sensores posibles proporcionan información en formatos muy diferentes, la predicción de la actividad se tiene que implementar usando algoritmos diversos. En [3] se proporciona una tabla resumen con varios de los métodos que han sido desarrollados hasta la fecha, que se estudiarán con más detalle en la sección correspondiente.

El reconocimiento de actividades humanas ha encontrado muchas aplicaciones. Este desafío se centra en un uso industrial, pero también existen aplicaciones relacionadas con la lucha contra el crimen [38], la moda virtual [14] o la salud [42]. Una correcta clasificación de las actividades que se realizan durante un proceso industrial por parte de los trabajadores permitiría posteriormente mejorar la eficiencia de ese proceso, al optimizar el número y el tipo de operaciones que se necesitan para el fin que se persigue.

1.1. Descripción del proyecto

El proyecto consiste en, utilizando el conjunto de datos proporcionado por los organizadores del desafío, realizar un desarrollo que permita investigar qué modelos son capaces de resolver mejor una tarea de Human Activity Recognition (en adelante, HAR), y en particular la presentada en el Open Pack Challenge 2022, para intentar crear una alternativa propia que sea competitiva. Uno de los principales problemas a los que se enfrenta cualquier desarrollo de modelos de aprendizaje automático, que pretenda aprender de un conjunto de datos que ha sido creado a partir de diversas fuentes de datos, es de qué manera aglutinar la información proveniente de las distintas fuentes. Ya que el conjunto de datos usado para el desafío tiene esas características, primero se realizará una exploración de las diferentes maneras que existen de combinar la información. Para ello, se desarrollaran varios modelos distintos que sigan una estrategia u otra de fusión. Dependiendo de las conclusiones que se saquen, se procederá a desarrollar un modelo más avanzado, pensado para participar en la competición. Las etapas en la ejecución del proyecto han sido las siguientes:

1. **Revisión del desafío y de las tecnologías utilizadas:** Esta fase consiste en familiarizarse con las características propias de la competición, como conocer qué tecnologías se han usado, cuál ha de ser el formato de la solución o qué herramientas proporcionan los organizadores.

2. **Estudio de los fundamentos teóricos, estado del arte y soluciones previas:** Implica la investigación acerca de en qué consisten exactamente las tareas de HAR, cuáles son las metodologías más aceptadas, y qué soluciones a problemas similares existen ya en la literatura. Además, dado que es un competición ya finalizada, el esquema general de las soluciones que quedaron entre los cinco primeros puestos está disponible públicamente, así como el código de la solución ganadora, con lo que es posible tener una idea general previa al comienzo del desarrollo de cuáles pudieran ser buenas soluciones.

3. **Estudio del conjunto de datos:** Para saber con qué tipos de datos se trabaja, como están estructurados y como realizar las lecturas de los ficheros proporcionados antes de empezar a codificar ningún modelo.

4. **Investigación en la fusión de modalidades:** Las mediciones que constituyen el conjunto de datos fueron obtenidas de sensores muy distintos, por lo que es necesario abordar

el problema de cómo es posible fusionar estas diferentes fuentes de datos (modalidades). Esta es ya una etapa de desarrollo, en la que se va a elegir un modelo teórico, y se van a implementar tantos modelos concretos como distintas estrategias de combinación de modalidades se investiguen.

5. **Desarrollo del modelo para la competición:** Los modelos desarrollados en el paso anterior proporcionan una idea de cuál podría ser una estrategia correcta de fusión. En este paso se procede a desarrollar un modelo tratando de que obtenga la máxima puntuación posible, siguiendo las indicaciones de la organización de cómo son evaluados los modelos participantes.

6. **Conclusiones y trabajo futuro:** Evaluación del resultado obtenido, análisis de los defectos en el desarrollo y de las posibles mejores, y propuestas para continuar investigando en esta línea.

7. **Redacción de la memoria.**

1.2. Objetivos

El objetivo del proyecto es en realidad un doble objetivo, con dos partes diferenciadas con cierta independencia entre sí:

1. **Comparación de estrategias de fusión de modalidades:** En esta parte se investigará en qué consiste la fusión de modalidades, qué clases de fusión existen, y cuáles pueden ser las ventajas o desventajas de cada una. El objetivo final de esta investigación previa es el desarrollo de varios modelos, cada uno implementando una técnica distinta. En esta caso han sido tres: fusión a nivel de sensor, de características y de decisión. Una vez desarrollados los modelos, se pueden comparar qué resultados se han obtenido en cada uno de ellos antes de continuar con el proyecto, y concluir cuál ha sido más efectivo. El objetivo en este paso no es, de ningún modo, obtener una puntuación alta en la clasificación, sino comparar las puntuaciones de los modelos entre sí, ya sean estas mejores o peores. Para no sesgar los experimentos, el modelo base a partir del cual se implementan las diferentes estrategias de fusión es el mismo en los tres casos, lo que permite restringir las diferencias en rendimiento entre uno u otro únicamente a las diferencias en la fusión de información.

2. **Participación en la competición:** Una vez que se determine qué estrategia de fusión es mejor para el conjunto de datos dado, empieza la segunda etapa, cuyo objetivo es maximizar la puntuación obtenida en la competición. Esta puntuación viene calculada por las propias herramientas que la organización proporciona. Los modelos que se desarrollen para este objetivo no tienen porqué tener nada que ver con los desarrollados en el paso previo, pues si aquellos eran elegidos por su potencial didáctico para discriminar entre estrategias de combinación de datos, ahora lo único que importa es implementar uno o varios modelos que realicen la clasificación de la mejor manera posible. Cabe decir que no es posible saber la puntuación que se hubiese obtenido de haber participado realmente, pues los modelos son evaluados con datos internos, pero se utilizará la puntuación que se obtenga sobre el conjunto de test oficial como una buena aproximación.

1.3. Estructura del documento

La memoria está organizada en siete capítulos:

1. **Introducción.**

2. **Fundamentos teóricos y estado del arte.** En este capítulo se explican los conceptos teóricos, tanto relacionados con Human Activity Recognition, como con el aprendizaje automático en general. También se reseñan las soluciones propuestas en la literatura para problemas similares.

3. **Herramientas y metodología.** Se especifica qué herramientas se utilizaron, tanto software como hardware, y cuál fue la metodología de desarrollo del proyecto.

4. **Exploración y análisis de datos.** Capítulo dedicado a explorar los datos proporcionados para resolver el desafío.

5. **Comparación de distintas estrategias de fusión.** Capítulo en el que se lleva a cabo el primero de los objetivos del proyecto: realizar una comparación entre las distintas maneras que existen de combinar modalidades. Para ello se elige un modelo que sea didáctico, y se evalúan los resultados.

6. **Desarrollo del modelo de la competición y evaluación de resultados.** Partiendo de los resultados del capítulo anterior, en este se implementa un modelo destinado a obtener la mejor puntuación posible. También se evalúan los resultados obtenidos, y se comparan con los de las soluciones que realmente participaron.

7. **Conclusiones.** Capítulo en el que se concluye el proyecto, y se detallan posibles líneas de mejora y trabajo futuro.

Capítulo 2

Fundamentos teóricos y estado del arte

2.1. Modalidad de los sensores en el reconocimiento de actividades humanas

Antes de comenzar a investigar qué propuestas existen para solucionar tareas similares a la presente, es necesario explicar una serie de conceptos importantes para el reconocimiento de actividades humanas. El primero de ellos es el de modalidad de los sensores. Se define modalidad como el tipo de información que pueden proporcionar distintos tipos de sensores, y que posteriormente será empleada para llevar a cabo la tarea de clasificación. Hay tantos tipos de modalidades como tipos de sensores existen, y/o como maneras tienen esos sensores de obtener datos acerca del mundo. Como ejemplos podríamos nombrar:

1. **Acelerómetro:** Mide la información de la aceleración a la que está sometida el sensor, normalmente presente en pulseras inteligentes y smartphones. Será utilizado en el presente proyecto.

2. **Giroscopio:** Mide la velocidad angular y permite obtener más precisión acerca de movimientos rotacionales que sólo el acelerómetro. También será usado en el presente proyecto.

3. **Micrófono:** Para obtener información acerca del sonido que pudiera diferenciar una actividad de otra.

4. **Cámara:** Información visual, en vídeo o imagen estática, que puede ser utilizada posteriormente en reconocimiento de imágenes

5. **Sensores de presión o temperatura:** Algunas actividades pueden generar mayor actividad corporal, y por tanto más temperatura que otras, y servir de criterio de clasificación. La presión puede ser importante en tareas realizadas en el exterior.

En [47] se realiza un estudio de las diferentes modalidades que se han venido utilizando hasta la fecha. Los autores destacan que, para capturar correctamente la naturaleza jerárquica de las actividades humanas, una clasificación basada exclusivamente en la dimensión temporal no es suficiente. Destacan que diferentes modalidades tienen distintas capacidades, concluyendo que las basadas en la visión son las más efectivas. También señalan que "los métodos multimodales tienen el potencial de reconocer actividades más detalladas, pero los métodos existentes no son capaces de abordar bien los datos multimodales"[47]. Esta última será una de las principales

investigaciones del proyecto: dado que el Open Pack Challenge ofrece en su conjunto de datos múltiples modalidades, será necesario encontrar una manera de fusionar los datos de aceleración, giroscopio, puntos clave... si se quiere desarrollar un mejor modelo.

2.1.1. Fusión de modalidades

Esta fusión de modalidades es importante, ya que actividades que proporcionen las mismos datos a un tipo de sensor pueden diferir en otro. Por ejemplo, una actividad dada puede tener los mismos datos de aceleración leídos por un sensor situado en la muñeca que otra, pero ambas diferir en las coordenadas de las extremidades inferiores, o viceversa. Para un modelo que sólo tuviera en cuenta una modalidad, esos cambios serían invisibles. Naturalmente, el grado de precisión y la cantidad de modalidades requeridas depende del propio problema de clasificación y de la complejidad de las actividades realizadas, pero en este caso se cumplen ambas condiciones: están disponibles en el conjunto de datos modalidades distintas (ya que los sujetos de pruebas llevaban puestos y estaban monitorizados por diferentes sensores, cada uno capaz de realizar lecturas de varios tipos distintos), y además las actividades son lo suficientemente complejas como para no ser fácilmente distinguibles entre sí, y requerir de toda la información adicional posible para su correcta clasificación.

Tomando como partida la clasificación dada en [23], la fusión de modalidades puede hacerse a tres niveles distintos:

1. **Fusión a nivel de sensor:** Cada una las de lecturas proporcionadas por los diferentes sensores en cada instante temporal se unen en una única medición, que constituirá una instancia para el posterior entrenamiento.

2. **Fusión a nivel de característica:** Se extraen características de mayor abstracción de cada una de las modalidades, ya sea mediante reglas codificadas a mano, o como resultado de un aprendizaje previo. Una regla codificado a mano podría ser algún tipo de métrica estadística, que fuese calculada sobre las muestras a entrenar, y que formara parte de un vector de características posterior. La alternativa sería dejar que el propio modelo aprenda esas características (mediante una red convolucional, por ejemplo). Estas características se combinan entre sí para formar una última instancia, que será la que finalmente sea clasificada.

3. **Fusión a nivel de modelo (decisión):** Cada modalidad se entrena de manera aislada. Una vez se ha completado el entrenamiento de tantos modelos como modalidades existan, se combinan entre sí para obtener la clasificación final, en la que cada modelo «vota» para asignar cada instancia a una clase. El peso de esos voto puede asignarse de manera estática (con pesos fijos para cada modelo), o como un meta-modelo que aprenda a asignar los pesos a cada uno de los clasificadores pre-entrenados, en función de como realicen la clasificación. A este tipo de fusión también se le llama habitualmente «por puntuación», porque lo que se hace es combinar las puntuaciones que cada uno de los modelos otorga a cada clase. En adelante se usarán ambos términos de manera indistinta.

Los métodos de fusión de modalidades presentan muchas similitudes con los métodos de combinación de modelos, en sus bases técnicas y marcos conceptuales. En [28] se realiza un estudio exhaustivo de cómo puede llevarse a cabo la combinación de clasificadores, en cualquiera de sus formas. Los autores ordenan los clasificadores en los mismos grupos: a nivel de sensor, de concatenación de características, o de decisión. Para esta última, distinguen a su vez tres tipos de

combinación: abstracta, por rango, y por puntuación. Según los autores, «En la abstracta, una única etiqueta de salida de cada clasificador individual es utilizada como entrada al esquema de combinación. Con las aproximaciones por rango, cada clasificador produce varias etiquetas, ordenadas de la más probable a la menos probable. Esta información es utilizada por el esquema de combinación para llegar a la conclusión final. La aproximación basada en puntuación (a veces llamada medida) es la técnica más informativa. Aquí, cada clasificador produce las mejores n etiquetas junto con su puntuación de confianza. La combinación de puntuaciones puede conseguirse de muchas maneras»[28].

El problema de la fusión a nivel de características o de sensores es que muchas veces, dependiendo de la fuente de datos, es muy complicado realizar la fusión, o sencillamente imposible. En [16], los autores entrenan un modelo para detectar minas de tierra, en los que utilizan sensores que son inconmensurables, por lo que necesitan desarrollar un algoritmo *ad-hoc* que permita la fusión a nivel de las características obtenidas por cada sensor. En [1], los autores se plantean el problema de qué tipo de fusión de modelo escoger en problemas de HAR: *stacking*, combinación de modelos, votación y Adaboost, por ser los más comunes. Desarrollaron un meta-algoritmo, que consiste en crear un vector de características a partir de propiedades estadísticas cada conjunto de datos completo, y luego comprobar qué modelo funciona mejor en ese conjunto de datos. Posteriormente, dado un nuevo conjunto de datos se puede utilizar como modelo de fusión aquél que ha funcionado en el conjunto de datos más parecido. En [8] generalizan esa idea (desarrollado inicialmente para HAR, y en concreto para sensores de aceleración y giroscopios) a cualquier conjunto de datos. En [7] utilizan la matriz de confusión de los diferentes decisores para obtener la clasificación final. Para lograr la fusión de modalidades visuales, acústicas y textuales en el reconocimiento de emociones, los autores de [46] utilizan una red neuronal en la última capa, que se entrena para elegir entre las etiquetas dadas por los clasificadores individuales. En [34], se realiza fusión a nivel de características en una tarea de clasificación de señales de radar, pero la fusión se realiza en el paso previo a la clasificación, con lo que también podría adoptarse una estrategia casi idéntica para realizar la fusión a nivel de decisión. Ma et al. [26] reseñan el estado del arte respecto a la fusión de modalidades para la clasificación de imágenes, y destacan que, a nivel de decisión, logran buenos resultados a coste de complejidad computacional. Dou et al. [12] proponen una fusión a nivel semántico para segmentar semánticamente una imagen: parten por un lado de imágenes RGB, y por otro de nubes de puntos obtenidas con un sensor *LiDAR*. Tras una extracción de características usando redes convolucionales, se genera una versión de la imagen en la que, por cada píxel, se da la probabilidad de que pertenezca a alguna de las clases. Una última capa llamada Vox-Net se encargará de fusionar esta información, y clasificar cada píxel en una clase. Wang et al. [43] desarrollo el método de *deep fusion*, por el cual las características, en lugar de fusionadas una única vez, son fusionadas iterativamente a través de varias redes neuronales. Por último, para terminar con este tema, como ejemplo de fusión pura a nivel decisión, en [48] se fusionaron dos sistema recomendadores: primero se obtenían las recomendaciones de cada uno de los modelos, y en el siguiente paso se ponderaban para obtener la recomendación final, siendo esta ponderación adaptativa, en función de los resultados del aprendizaje.

2.1.2. Tasa de muestreo de diferentes modalidades

Uno de los principales problemas para la fusión de modalidades es que pueden tener tasas de muestreo diferentes. La tasa de muestreo, o frecuencia de muestreo, se refiere al número de muestras o mediciones que se lee un sensor por unidad de tiempo. La unidad de medida son los

hercios (Hz), que son inversamente proporcionales al tiempo que se tarda en medir. Es decir, una tasa de muestreo de 10Hz quiere decir que el sensor lee 10 veces cada segundo. En el caso del presente proyecto, los tres sensores que se usarán (y que se explican más adelante) son el E4, el ATR-TSND151 y el Microsoft Kinect, con tasas de muestreo de 32, 30 y 15 Hz respectivamente. Como se ilustra en la figura 2.1, esto presenta el problema de que, para un segundo dado, la cantidad de muestras que han tomado cada uno de los sensores es distinta.

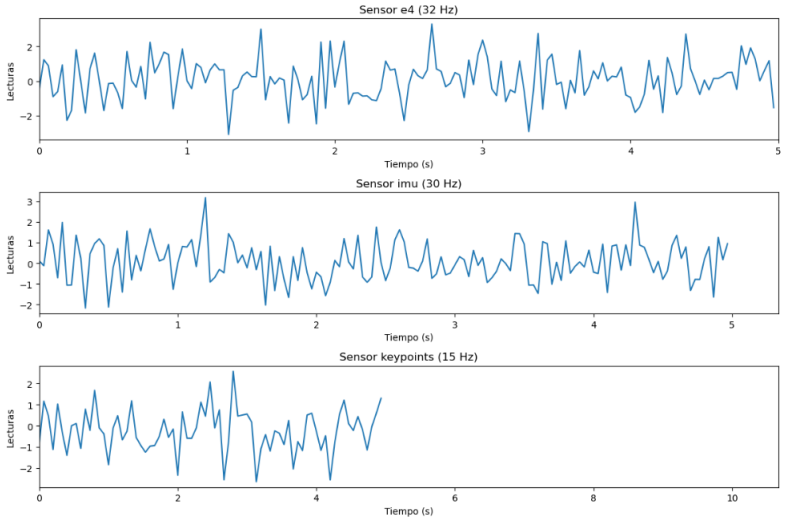

Figura 2.1: Ejemplo de lecturas de los tres sensores. Datos únicamente ilustrativos

En algún momento se ha de utilizar el *ground truth* con las etiquetas, y como estas corresponden a cada instante de tiempo, la dimensionalidad del vector de etiquetas será distinta para cada ventana de tiempo dada. Por ejemplo, si queremos fusionar los tres sensores para clasificar las actividades que se realizaron en un segundo específico, el sensor e4 obtendrá 1920 lecturas (y esa será la dimensionalidad de su vector de etiquetas), el sensor IMU obtendrá 1800 lecturas, y Kinect obtendrá 900. Calcular la función de pérdida así es imposible, con lo que existen varias estrategias: hacer padding, añadiendo con un dato fijo los lugares faltantes, interpolar, o resamplear todo al valor más bajo. En [24], por ejemplo, optan por interpolar los datos faltantes.

2.2. Clasificación de series temporales y secuencias

La tarea de clasificar correctamente una actividad humana puede ser englobado de manera más general dentro de la clasificación de series temporales, así que se explicará brevemente en qué consiste esta, y se darán algunos ejemplos. La clasificación de series temporales es un tipo de aprendizaje automático que se caracteriza por trabajar con datos que están ordenados temporalmente. En realidad, que el ordenamiento sea estrictamente temporal no es relevante. De acuerdo con [4], «la característica importante es que existan características discriminatorias en función del ordenamiento». Habitualmente, en clasificación de series temporales se entiende que el objetivo es clasificar cada instancia $\mathbf{x}$, compuesta de m observaciones, en su clase correspondiente y. Para este desafío, sin embargo, lo que piden los organizadores es que los modelos sean capaces de clasificar individualmente cada una de las mediciones realizadas por los sensores. Es decir, que la salida del modelo a partir de la secuencia de entrada $\mathbf{x}$, en lugar de ser una clase y, será otra secuencia de igual longitud $\mathbf{y}$, con cada una de las predicciones de a qué clase corresponde cada elemento de la secuencia.

La clasificación de series temporales se puede aplicar a numerosos problemas diferentes: clasificación o traducción de texto, análisis de secuencias de ADN, predicción bursátil... Dependiendo de la estructura de los datos que deseemos clasificar, algunos algoritmos han demostrado desempeñar mejor que otros. En [15], se comparan varios modelos, tanto de aprendizaje automático tradicional como de deep learning, para predecir el precio de cierre de la jornada bursátil, y encontraron que en general modelos basados en deep learning tenían una mejor precisión para esa tarea. También en [11] se utiliza el deep learning para la predicción del mercado bursátil indio, y se comparan varios modelos entre sí. La conclusión es que los modelos basados en LSTM y GRU obtenían el mejor rendimiento, y que el modelo basado en Temporal Fusion Transformer podía obtener un rendimiento casi tan alto, pero no mejor. En [20] se comparan también algoritmos de aprendizaje automático tradicional con deep learning, en este caso para la predicción de la demanda, y apreciaron que si la cantidad de datos usado para el entrenamiento es reducida, los algoritmos basados en deep learning ven afectado su rendimiento.

En conclusión, la clasificación de series temporales puede abordarse desde una amplia variedad de algoritmos y metodologías. En los siguientes apartados, veremos cuáles han sido los más frecuentes en la clasificación de actividades humanas según la literatura, y cuáles han sido los que de hecho fueron utilizados en la competición, para posteriormente justificar la elección de un modelo.

2.3. Fundamentos de Human Activity Recognition

Debido a que los seres humanos se ven involucrados en una gran diversidad de actividades distintas, también las aproximaciones para lograr la clasificación de estas son muy diversas. Dependiendo de qué clase de actividades queramos clasificar, o cuál sea el propósito de la clasificación, se empleará una metodología u otra.

En [2] se hace un estudio pormenorizado de la literatura existente, y se clasifican los trabajos previos en función de las áreas de aplicación y de las técnicas empleadas. Los autores distinguen entre actividades del día a día, actividades en tiempo real, y actividades en grupo.

También en el mismo trabajo, los autores desglosan los algoritmos que han sido utilizados en la literatura. En la figura 2.3 se muestran sus resultados. Como se ve, las redes convolucionales, y las de tipo LSTM dominan los trabajos realizados, arquitecturas que serán explicadas en la siguiente sección.

Con respecto a los retos a los que se enfrenta, en el mismo estudio se enumeran cinco:

1. **Recolección de datos:** Existen datos sin enumerar, mal alineados, incorrectamente etiquetados,etc... Recoger datos, especialmente si es de actividades del día a día, puede ser una tarea compleja.

2. **Preprocesamiento de datos:** Como limpiar el trasfondo de la leutra de los sensores, o añadir características configuradas a mano a un modelo de deep learning para aumentar su precisión.

3. **Hardware y técnicas:** El hardware puede ser el cuello de botella para algunos algoritmos. En particular, la extracción de características que suele realizarse mediante redes convolucionales es muy exigente de computar.

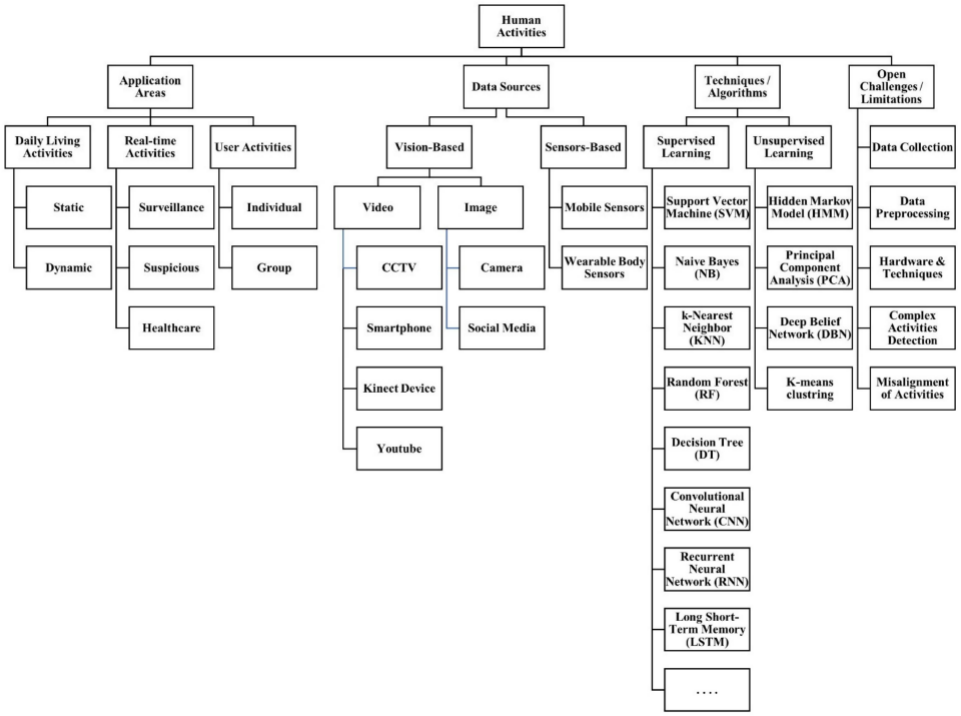

Figura 2.2: Diagrama clasificatorio de HAR. Fuente original: [2]

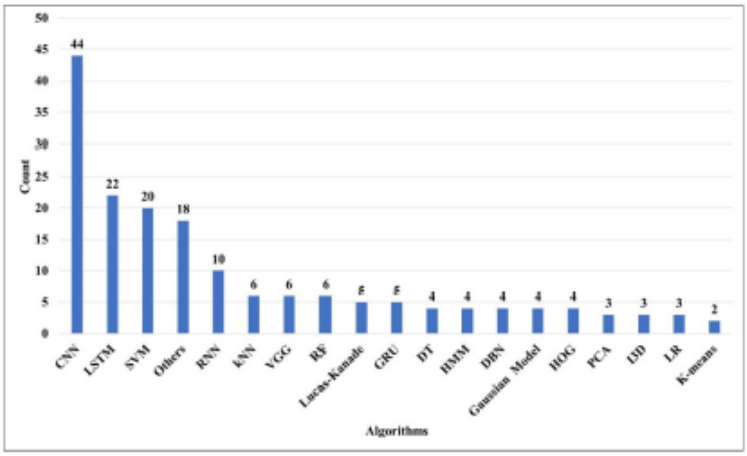

Figura 2.3: Técnicas empleadas en HAR. Fuente original: [2]

4. **Detección de actividades complejas:** Algunas actividades (como cocinar) están compuestas de numerosas acciones individuales sencillas. Es necesario un reconocimiento concurrente de actividades para integrar múltiples acciones sencillas.

5. **Actividades desalineadas:** La anotación manual de datos de esta clase consume mucho tiempo, y utilizar datos mal anotados impide que se alcancen niveles altos de precisión.

Gupta et al. [17] describen cuatro etapas en el proceso de HAR:

1. Capturar la señal de una actividad.

2. Preprocesar los datos.

3. Reconocer la actividad.

4. Interfaz de usuario para gestionar el proceso.

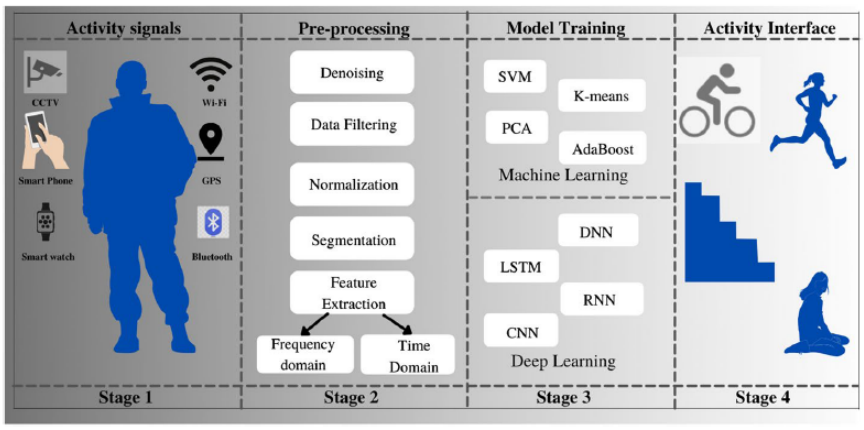

Figura 2.4: Esquema de las etapas en HAR. Fuente original: [17]

En resumen, el reconocimiento de actividades humanas es un tipo de tareas de clasificación que se enmarcan dentro del aprendizaje automático, y más concretamente en la clasificación de series temporales, en los que uno o varios individuos realizan una serie de actividades que deben ser clasificadas. Los datos que se quieren clasificar pueden provenir de fuentes de muy diverso tipo, pero destacan fundamentalmente los sensores portables (como pulseras electrónicas), y las capturas de vídeo. Uno de los principales retos es precisamente el desarrollar métodos de combinar las fuentes de datos entre sí, aprovechando la mayor información que se obtiene cuando la misma actividad es grabada desde diferentes sensores a la vez. Obtener un conjunto de datos anotado se vuelve más complicado a medida que aumenta la complejidad de los sensores, la situación en la que ha tenido lugar la grabación, la longitud temporal que tomó llevar a cabo la grabación (a mayor duración, mayor probabilidad de que se produzcan desalineamientos, o que algunos segmentos estén mal etiquetados, etc...). Para evitar estos problemas en los datos cuando se intentan evaluar los distintos modelos entre sí, existen numerosos conjuntos de datos públicos disponibles.

2.3.1. Conjuntos de datos habituales para la evaluación de HAR

Antes de explicar con detalle en secciones posteriores los datos que se utilizan en el Open Pack Challenge, describo brevemente algunos conjuntos de datos que aparecen con frecuencia en la literatura:

1. **WISDM:** WISDM es un conjunto de datos elaborado por el laboratorio Wireless Sensor Data Mining en la universidad de Fordham. Descrito con más detalle en [44], contiene datos de recogidos del acelerómetro y giroscopio de telefónos móviles y relojes inteligentes por parte de 51 sujetos distintos que realizaron 18 actividades distintas.

Actividad	Código
Caminar	A
Correr	B
Escaleras	C
Sentarse	D
De pie	E
Escribiendo en teclado	F
Cepillando los dientes	G
Tomando sopa	H
Comiendo patatas de bolsa	I
Comiendo pasta	J
Bebiendo de una copa	K
Comiendo un sandwich	L
Pateando un balón de fútbol	M
Jugando al tenis	O
Regateando (baloncesto)	P
Escrbiendo a mano	Q
Aplaudiendo	R
Doblando ropa	S

Cuadro 2.1: Tabla con las actividades que forman parte del conjunto de datos WISDM

2. **MHealth:** Datos descritos en [5]. Las lecturas se llevaron a cabo mediante los sensores portables Shimmer2, y participaron 10 sujetos de distintos perfiles que realizaron 12 actividades distintas:

Actividad	Código
Estar de pie	L1
Sentado, relajado	L2
Tumbado	L3
Caminando	L4
Subiendo escaleras	L5
Doblando el tronco	L6
Levantando los brazos	L7
Doblando las rodillas	L8
Pedalear	L9
Andar	L10
Correr	L11
Saltar	L12

Cuadro 2.2: Tabla con las actividades que forman parte del conjunto de datos MHealth

3. **PAMPA2:** Conjunto de datos creada a partir de 18 actividades físicas distintas, llevadas a cabo por 9 sujetos diferentes, y medidas con 3 medidores inerciales, y un medidor cardíaco [32].

Actividad	Código
Tumbado	1
Sentado	2
De pie	3
Caminando	4
Corriendo	5
Pedaleando	6
Marcha nórdica	7
Mirando la TV	9
Trabajando en el ordenador	10
Conduciendo	11
Subiendo escaleras	12
Bajando escaleras	13
Aspirando	16
Planchando	17
Doblando la ropa	18
Limpiando la casa	19
Jugando al fútbol	20
Saltando a la comba	24

Cuadro 2.3: Tabla con las actividades que forman parte del conjunto de datos PAMPA2

Estos son tan solo algunos ejemplos, podrían darse muchos más. En general, todos los conjuntos de datos se crean realizando una serie de mediciones (con diversos dispositivos, pero los acelerómetros suelen ser muy frecuentes por su sencillez de uso), sobre una serie de sujetos que desempeñan unas actividades dadas. En estos ejemplos, las actividades son muy generales, y muy distintas entre sí. Como se expondrá más adelante, las actividades que componen el Open Pack Challenge son mucho más parecidas, no implican diferencias de movimientos tan acusadas (como las que implican, por ejemplo, entre estar sentado y corriendo), y por tanto son más difíciles de clasificar.

2.4. Fundamentos de Deep Learning

En este apartado se proceden a explicar diferentes arquitecturas y conceptos asociados con el deep learning que se han utilizado en el desarrollo de los distintos modelos que componen el resultado final del proyecto. Aquí se detallan las bases teóricas generales, posteriormente se explicará de qué manera se ha utilizado cada red.

2.4.1. Redes convolucionales

Las Redes Neuronales Convolucionales (CNN, por sus siglas en inglés, Convolutional Neural Networks) son un tipo de red neuronal artificial diseñada específicamente para el procesamiento de datos estructurados en forma de matriz, como imágenes y vídeos. Se utilizan principalmente en tareas de visión por computador, reconocimiento de patrones y procesamiento de imágenes debido a su capacidad para capturar y aprender características relevantes de los datos. Cuando dentro de una misma red se incluyen múltiples capas convolucionales, estaríamos hablando de

una arquitectura de red convolucional profunda (DCNN, Deep Convolutional Neural Network). Normalmente, en una red de estas características aparecen también capas de agrupación (*pooling*) que permiten reducir la dimensionalidad de los datos, y capas completamente conectadas, habitualmente a la salida de la red para llevar a cabo la clasificación. El apilar numerosas capas convolucionales en una misma red permite aprender las características de los datos en grados progresivos de mayor abstracción, de manera jerárquica y automática.

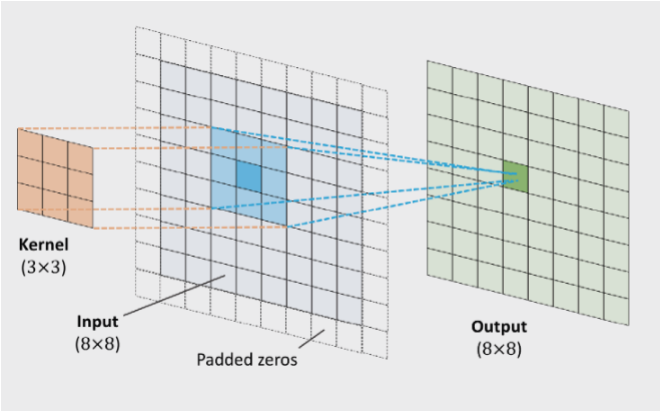

Figura 2.5: Ejemplo de una capa convolucional. Fuente original: [31]

Las redes convolucionales ganaron popularidad debido a su éxito clasificando imágenes, aunque sus aplicaciones transcienden el reconocimiento visual. A medida que estas redes demostraban ser exitosas en problemas cada vez más complejas, se iban proponiendo diversas arquitecturas, que varían en cuanto al profundidad de la red, el tipo de capas convolucionales o el tratamiento de los datos a clasificar, y que han tratan de superarse unas a otras. Podemos mencionar como ejemplos significativos GoogleLeNet [39], ResNet [18] o VGGNet [36] por su relevancia histórica, pero existen muchas más. Un tipo de arquitectura especialmente relevante son las *Fully Convolutional Network* (FCN) [35]. Este tipo de redes prescinden de la última capa completamente conectada, que era la encargada de llevar a cabo la clasificación, sustituyéndola por una red convolucional adicional. Este pequeño cambio permite sin embargo lograr flexibilidad en el tamaño de las imágenes (o en general, de los datos de entrada) que se quieran clasificar.

Dentro de las FCN, la arquitectura U-Net [33] destaca ampliamente por su popularidad para lograr segmentación semántica de imágenes, es decir, asignar a cada píxel de una imagen la clase que le corresponde. Este es un problema similar al del presente proyecto, en el que también se trata de asignar una clase a cada elemento de la secuencia de acciones que se quiere clasificar. La arquitectura UNet consiste en una sucesión de capas convolucionales, que pueden incluir normalización por lotes y *dropout*, y que forman la parte de codificador (*encoder*) seguidas de una secuencia de capas de convolución traspuesta, que forman la parte del decodificador (*decoder*). El propósito de la parte del codificador es reducir la dimensionalidad de los datos de entrada aplicando capas convolucionales sucesivas, mientras se aumenta el número de caracerísticas, obligando a la red a aprender los aspectos más relevantes de los datos. La parte del decodificador parte de la información comprimida, y aumenta gradualmente la resolución de las características aprendidas por el codificador.

2.4.2. Redes recurrentes y LSTM

Las redes neuronales recurrentes (RNN, Recurrent Neuronal Network) se diseñaron para trabajar con datos que presentan dependencias temporales. La diferencia con una red convencional es que la presencia de conexiones recurrentes les permite mantener información previa, y utilizarla para procesar datos futuros dentro de una secuencia. Gracias a mantener esta memoria, las redes recurrentes son capaces de aprender relaciones entre datos separados temporalmente. Uno de los usos más habituales de las RNN ha sido el procesamiento del lenguaje, que presenta de forma natural un ordenamiento temporal, y por supuesto para HAR, estando entre las arquitecturas que aparecen con más frecuencia en la literatura.

Un problema que afecta a este tipo de redes es el del desvanecimiento/explosión del gradiente, debido al cual se complica el aprendizaje de patrones si implican datos que estén muy separados temporalmente, ya que el gradiente tiende a cero o a infinito a medida que se retropropaga entre ambos. Una de las posibles soluciones es el uso de capas LSTM (Long Short Term Memory), variente de las RNN que está diseñada para ser capaz de capturar este tipo de dependencias a largo plazo. Una unidad LSTM está compuesta de las siguientes partes:

1. **Puerta de olvido:** decide qué información almacenada en el paso anterior debe olvidarse, y cuál mantenerse para el paso actual.
2. **Puerta de entrada**: determina qué nueva información agregar a la célula de memoria.
3. **Celda de memoria:** Almacena y actualiza la información a lo largo del tiempo.
4. **Puerta de salida:** Decide qué información se debe enviar como salida en el momento de tiempo actual.
5. **Salida:** Salida final de la unidad, tanto a la siguiente unidad LSTM en una secuencia, o como predicción final.

Las ecuaciones para el cálculo de cada una de las partes se encuentran, por orden, a continuación:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{2.1}$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{2.2}$$

$$\tilde{C}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \tag{2.3}$$

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \tag{2.4}$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{2.5}$$

$$h_t = o_t \odot \tanh(C_t) \tag{2.6}$$

2.4.3. Auto-atención

Los mecanismos de atención son una alternativa a las RNN para trabajar con secuencias de datos. A diferencia de las RNN, la atención permite a una red neuronal artificial acceder a todos los elementos de una secuencia al mismo tiempo. Para filtrar parte de toda esa información, que al pertenecer a la secuencia completa se pueda hacer inabarcable, un mecanismo de atención asigna un peso a los elementos de entrada. Estos pesos desvían la atención del modelo, que se centra en aquellos con un valor mayor (de ahí el nombre de este tipo de algoritmos).

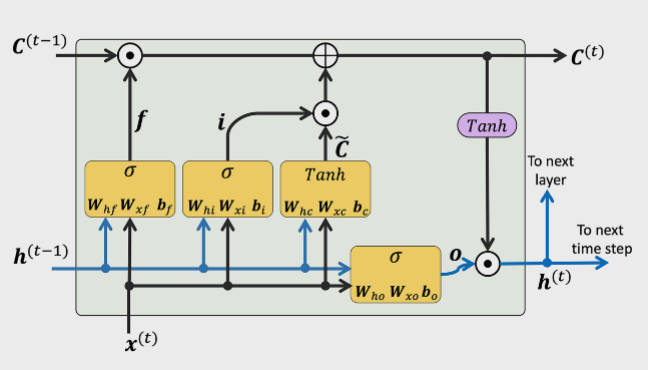

Figura 2.6: Estructura de una unidad LSTM. Fuente original: [31]

Dentro de los posibles mecanismos de atención, hay uno que ha cobrado mucha relevancia: la auto-atención. Especialmente utilizada en tareas de procesamiento de lenguaje natural, permite que un modelo procese secuencias de entrada considerando las relaciones entre todas las posiciones de la secuencia entre sí, es decir, como se relacionan entre ellos los elementos de una secuencia. Sin entrar en muchos detalles, se utilizan tres matrices de pesos (*query*, *key*, *value*), que se multiplican por cada uno de los elementos de entrada para obtener sus respectivas secuencias. Estas secuencias son las que permiten calcular en la siguiente etapa los valores de los pesos de atención. En la figura 2.7 se muestra un esquema del funcionamiento:

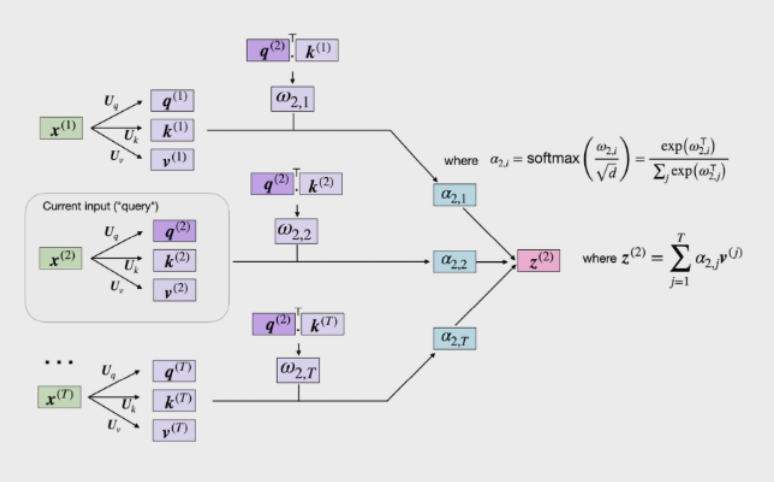

Figura 2.7: Esquema del funcionamiento del mecanismo de autoatención. Fuente original: [31]

2.5. Deep Learning aplicado a Human Activity Recognition

Ahora que están explicados varios conceptos fundamentales relativos al deep learning, se procede a reseñar el estado del arte respecto a sus usos para HAR. Debido a la naturaleza secuencial de las actividades humanas, los tres tipos de redes descritos anteriormente (CNN, RNN y atención) aparecen con frecuencia en la literatura. Habitualmente no aparecen aislados, sino como parte de arquitecturas más elaboradas, que suelen incluir varios de estos modelos combinados entre sí. Los principales motivos del uso de estos modelos son:

1. **CNN**: Las redes convolucionales son capaces de extraer características de alto nivel, respetando la estructura a nivel local de los datos. Si presentamos una secuencia completa al modelo, esté será capaz de aprender patrones que existan entre los diferentes elementos entre sí. Es decir, que las redes de tipo CNN (en esta aplicación en concreto) ven la secuencia completa que corresponde a una ventana de tiempo dada como çongelada en el tiempo", toda a la vez, y a partir de ahí extraen características de mayor nivel de abstracción. En ocasiones, se utilizan algoritmos para transformar una secuencia de datos en una imagen, y realizar la clasificación partiendo de ahí.

2. **RNN:** Las redes recurrentes son la forma más natural de modelar datos secuenciales, por lo que su uso también es muy común en el contexto de HAR. Mientras que las CNN se usan para obtener aracterísticas de más alto nivel que abstraigan las particularidades de los datos, las RNN se utilizan para captar relaciones temporales entre los datos, es decir, dependencias en el orden de estos. Cuando se combinan ambas, habitualmente una serie de capas CNN extrae características, y una segunda etapa con redes recurrentes aprende las relaciones temporales entre estas. Prácticamente siempre son de tipo LSTM.

3. **Mecanismos de atención:** Han ganado popularidad en el deep learning en los últimos años, tanto con mecanismos de atención simple como utilizando la arquitectura de tipo Transformer, descrita en [41], y que consiste en una forma más compleja de auto-atención, en la que se intercalan varias unidades codificador-decodificador. Este tipo de redes son básicamente otra manera de implementar el aprendizaje de las relaciones temporales entre elementos distinta de las RNN, y por tanto se utilizan con propóitos parecidos.

En [30] se propone una arquitectura que combina primero redes convolucionales con redes recurrentes de tipo LSTM en un segundo paso. Lo relevante e este trabajo fue el sustituir la última red completamente conectada por las unidades LSTM, y ha servido de base para muchos otros trabajos. De hecho, es el que se menciona como ejemplo para tener un *baseline* en el propio repositorio del *Open Pack Challenge*. Singh et al [37] utilizan una arquitectura que combina los tres tipos de redes: primero, varias capas convolucionales aprenden las características de los datos, posteriormente varias unidades LSTM aprenden las relaciones temporales, y una última capa de auto-atención se utiliza para aprender las relaciones entre puntos temporales en los datos provenientes de los sensores. En [6], se utilizan capas convoluciones en un primer paso, y posteriormente una fusión de características a partir de la salida de redes recurrentes. Essa y Abdelmaksoud [13] utilizan un modelo sin redes recurrentes que denominan CSNet, en el que se alternan capas convolucionales y capas de auto-atención, hasta que una última capa completamente conectada realiza la clasificación. En [9], los autores utilizan un modelo que combina capas de *conformer*: denominan así a la combinación de capas convolucionales con capas de transformer. Estas capas de conformer se utilizan para extraer las características de los datos, que siendo este caso de reconocimiento de voz, vienen en dos modalidades: el audio de la voz, y el vídeo de la persona hablando. Posteriormente, concatenan las características extraídas antes de la clasificación final. En [25] se utiliza simplemente una arquitectura de Transformer, sin ninguna otra capa. Kavi et al [21] concatenan redes convolucionales y completamente conectadas, hasta llegar a una última capa con unidades LSTM. Por último, tanto los autores de [19] como de [10] utilizan redes convolucionales y unidades LSTM, en ambos además se bifurcan los datos a clasificar en varias ramas de procesamiento, para en la siguiente etapa fusionarse las características obtenidas por cada una de las ramas antes de la clasificación final.

2.5.1. Conclusiones

De un total de 11 arquitecturas reseñadas, 10 utilizan redes convolucionales en algún punto, 7 utilizan redes recurrentes, y 4 utilizan mecanismo de atención de algún tipo. Respecto al tipo de fusión de datos que se realiza, 2 lo hacen a nivel de sensor, 4 a nivel de característica, 1 a nivel de decisión/puntuación, y 4 no realizan fusión de ningún tipo. Nótese que se ha restringido la reseña de trabajos a aquellos en los que el problema a resolver fuera casi idéntico al del presente proyecto. Un estudio más exhaustivo podría arrojar resultados diferentes.

2.6. Medida de evaluación

El ranking de soluciones de *Open Pack Challenge* se elabora a partir de la medida F1 de las soluciones de los participantes. Las medidas tipo F son un tipo de métricas que se utilizan para evaluar modelos de clasificación, especialmente útil si las clases están desbalanceadas, es decir, que alguna sea mucho más frecuente que otra. Se basa en combinar otras dos métricas individuales: la precisión y la cobertura.

1. **Precisión:** Mide la proporción de predicciones positivas correctas, dividida entre el total de predicciones positivas (verdaderos positivos más falsos positivos)

   $P = \frac{VerdaderosPositivos}{VerdaderosPositivos+FalsosPositivos}$

2. **Cobertura**: Mide la proporción de predicciones positivas correctas, dividida entre el total de ejemplos positivos (verdaderos positivos más falsos negativos)

   $C = \frac{VerdaderosPositivos}{VerdaderosPositivos+FalsosNegativos}$

La clasificación F-1 es una medida específica de F, en la cual se combinan la precisión y la cobertura asignándoles el mismo peso:
$F - 1 = \frac{2 \cdot P \cdot C}{P+C}$

2.7. Soluciones previas del Open Pack Challenge 2022

Debido a que la competición había finalizado antes de comenzar este trabajo, es posible tener en cuenta qué soluciones han sido aportadas por los participantes. La arquitectura de las cinco que han quedado en mejor posición ha sido publicada por los organizadores. Además, las soluciones que alcanzaron las posiciones primera y tercera también han publicado su código, compartiendo el enlace de GitHub del proyecto.

La tabla 2.4 es un resumen, utilizando las bases teóricas desarrolladas a lo largo del capítulo, de estas soluciones. Todas utilizan redes convolucionales para aprender características locales en los datos. La mayoría de las soluciones también utiliza unidades LSTM, excepto precisamente la ganadora. Respecto a la fusión de modalidades, la solución ganadora implementó la fusión a nivel de características, la segunda tanto en características como en puntuación y la tercera solo en puntuación. El resto no implementaron fusión de modalidades porque utilizaron datos de sólo un sensor.

Equipo	Modalidades utilizadas	CNN	LSTM	Atención	Fusión	F1
tomoon	IMU, puntos clave, IoT logs, *bounding box*	Sí	No	Sí	Características	0.9633
vbu211	IMU, puntos clave, escáner manual, profundidad de imágen	Sí	Sí	No	Características y puntuación	0.9592
Ritsumei	IMU, puntos clave	Sí	Sí	No	Puntuación	0.9241
Malton	IMU	Sí	Sí	No	Sin fusión	0.9171
Shubham Wagh	IMU	Sí	Sí	Sí	Sin fusión	0.9112

Cuadro 2.4: Tabla resumen de las soluciones previas al Open Pack Challenge 2022

Capítulo 3

Herramientas y metodología

3.1. Herramientas software

El desarrollo se ha efectuado completamente en el lenguaje de programación *Python*. Esto es así porque, además de ser un estándar para trabajar en ciencia de datos, la competición estaba organizada para que las soluciones se entregasen utilizando el framework para deep learning *Pytorch*. Naturalmente, ya que no se va a participar realmente, se podría haber escogido cualquier otro framework (por ejemplo, *Tensorflow* en caso de que se quisiera que la solución siguiese siendo de deep learning), pero la organización proporciona algunas librerías bastante útiles, que incluyen lectura del conjunto de datos, modelos base con los que empezar a trabajar, y evaluación de las soluciones desarrolladas. Debido a esto, al elegir *Pytorch* puedo sacar provecho del código ya desarrollado por los organizadores sin tener que implementar la infraestructura del proyecto yo mismo. En el resto de la sección se describen las herramientas más importantes de entre todas las que se han utilizado.

3.1.1. Herramientas de desarrollo

El código se ha escrito utilizando *Visual Studio Code*[1], que es editor de código gratuito y de código abierto desarrollado por Microsoft, y que proporciona un amplío soporte para *Python*. Para el control de versiones se ha utilizado *GitHub*[2], una plataforma on-line para la colaboración y gestión de versiones. Todo el código realizado para el proyecto se puede encontrar en el repositorio [3].

3.1.2. Herramientas genéricas de ciencia de datos

Pytorch[4] es una librería de aprendizaje automático para *Python*, basada en la librería *Torch*[5], que se utiliza fundamentalmente en el campo del deep learning, y que destaca por su grafo de computación dinámico. *Lightning* [6] es un framework para *Pytorch*, que proporciona una interfaz de alto nivel que abstrae de los detalles del proceso de organizar y entrenar modelos en *Pytorch*,

[1] https://code.visualstudio.com
[2] https://github.com/
[3] https://github.com/enpe693/tfm1/
[4] https://pytorch.org
[5] http://torch.ch
[6] https://lightning.ai/

con objeto de simplificarlo. *Pandas* [7] es una librería para Python que ofrece estructuras de datos que son muy eficientes para almacenar, limpiar o manipular conjuntos de datos grandes. *NumPy*[8] es una librería de aprendizaje automático para Python que permite trabajar con arrays multidimensionales. *Tensorboard*[9] es una herramienta de visualización, que permite monitorizar gráficamente el entrenamiento y la evaluación de un modelo

3.1.3. Herramientas específicas de Open Pack Challenge

Open Pack ToolKit[10] proporciona una serie de utilidades para cargar el conjunto de datos del desafío, para gestionar los conjuntos de entrenamiento, validación y test, así como para evaluar la precisión de la solución creada. *Open Pack Torch*[11] incluye modelos ya desarrollados en *PyTorch*, para que puedan servir como guía, clases que permiten la carga de los datos en memoria para entrenar el modelo o tutoriales de cómo realizar el desarrollo del modelo y su empaquetado para presentarlo en la competición, entre otras utilidades.

3.2. Herramientas hardware

Todo el desarrollo se ha hecho en local, utilizando un ordenador personal que incluye GPU NVIDIA de 8GB de RAM.

3.3. Metodología

La metodología empleada para el desarrollo está basada en CRISP-DM [45], un modelo de procesos para proyectos en minería de datos, aplicable en general a los proyectos en ciencia de datos y aprendizaje automático.

En la figura 3.1 se muestra cómo los autores de CRISP-DM conciben las diferentes fases del desarrollo de un proyecto de minería de datos. Aplicado al proyecto presente, en cada una de las fases se han llevado a cabo las siguientes tareas:

1. **Comprensión del negocio.** Fase inicial que se centra en los requisitos del proyecto, y dónde se definen los objetivos en tanto proyecto de minería de datos. En este caso, el proyecto en sí mismo es una aplicación directa de la ciencia de datos, al consistir en el desarrollo de modelos de aprendizaje automático. El objetivo (obtener una puntuación alta en el desafío) no tiene unos requisitos de negocio previos.

2. **Comprensión de los datos.** Descarga del conjunto de datos proporcionado, exploración, evaluación de qué clases hay que clasificar, cómo se han recogido los datos, que problemas puedan existir (como datos ausentes, etc...). El objetivo final es la construcción de un conjunto de datos final, ya preparado para que el modelo sea entrenado en él.

3. **Modelado.** Dependiendo de qué clase de datos se tengan, si hay limitaciones en capacidad computacional o tiempo, o el propósito de la clasificación, se optará por un modelo u otro.

[7] https://pandas.pydata.org/
[8] https://numpy.org/
[9] https://www.tensorflow.org/tensorboard/
[10] https://github.com/open-pack/openpack-toolkit
[11] https://github.com/open-pack/openpack-torch

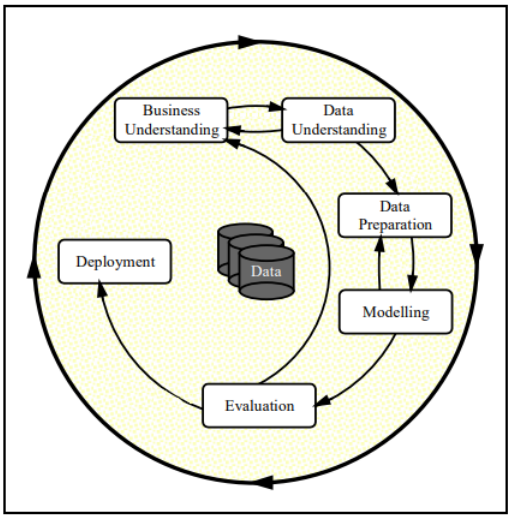

Figura 3.1: Ciclo de un vida de un proyecto de minería de datos. Fuente original: [45]

En este caso, esta etapa consiste en reseñar la literatura existente y las soluciones previas para decantarse por un modelo, e implementarlo.

4. **Evaluación.** Con el modelo ya diseñado, se procede a su entrenamiento, y posterior evaluación. Se toman medidas correctivas en caso de ser necesario, por ejemplo si su rendimiento es peor del esperado. En este caso, la evaluación se lleva a cabo utilizando el conjunto de test proporcionado junto con los datos, lo que permite dar una idea de cómo sería el desempeño real del modelo.

5. **Despliegue.** Cuando se tiene la suficiente certeza de que el modelo desarrollado va a lograr un buen rendimiento, es hora de ponerlo en producción. No es necesario para este proyecto, pues no se van a consumir externamente los resultados producidos por el modelo.

Capítulo 4

Exploración y análisis de datos

4.1. Estructura del conjunto de datos

Tal y como se explicó anteriormente, los datos acerca de las actividades humanas han sido capturados sobre 16 sujetos, por parte de 8 sensores diferentes, y anotados en 10 operaciones distintas. Cada una de las operaciones está desglosada a su vez en anotaciones acerca de las acciones que corresponden a una acción. Por ejemplo, la operación «trasladar la etiqueta del objeto» se descompone en acciones como «quitar etiqueta del objeto», «colocar etiqueta en el borde inferior», «comprobar datos de la etiqueta», etc...

Cada una de las distintas partes del conjunto de datos se explica con más detalle posteriormente.

4.2. OmegaConf

A pesar de que es posible descargar los datos manualmente, y trabajar con los archivos que contiene directamente, la organización del desafío los entrega utilizando OmegaConf. OmegaConf [29] es un sistema de configuración jerárquico, que permite mantener organizados archivos de configuración que provengan de diversas fuentes. En este conjunto de datos se proporciona una API para acceder a los datos organizados por sensor, usuario o sesión, abstrayendo la estructura subyacente. Es la forma recomendada por la organización, y la que se seguirá para el desarrollo del trabajo.

4.3. Anotaciones

Las anotaciones se corresponden con las etiquetas que son el objetivo de la clasificación. Hay dos tipos de anotaciones: operaciones, que describen acciones generales del usuario, y acciones, que describen los pasos que se dan par llevar a cabo una operación. Cada operación se descompone en acciones. El modelo tomará como entrada un conjunto de datos, y tratará de asignar la operación o actividad adecuada. En las siguientes tablas se resumen los posibles valores que pueden tomar, junto con su explicación.

4.3.1. Operaciones

ID	Operación	Descripción	ignorar
100	Recoger	Comprobar la hoja de orden, y recoger los objetos	No
200	Trasladar etiqueta del objeto	Arrancar la etiqueta del objeto, y situarla en la parte de abajo de la hoja de orden. Comprobar los nombres de los objetos y las cantidades en la lista, y marcarlos con un rotulador.	No
300	Ensamblar caja	Ensamblar las cajas de cartón que se correspondan con el tamaño de los objetos.	No
400	Insertar objeto	Poner los objetos en la caja, y rellenar el hueco con almohadillas	No
500	Cerrar caja	Cerrar la caja con cinta.	No
600	Pegar etiqueta a la caja	Poner la etiqueta numérica de la caja en un lado.	No
700	Escanear etiqueta	Escanear los códigos de barras con el escáner manual.	No
800	Pegar etiqueta de envío	Pegar la etiqueta imprimida con la impresora en la caja	No
900	Colocar en la mesa trasera	Mover el paquete completo a la mesa trasera	No
1000	Cumplimentar la orden	Escribir la firma en la columna de confirmación de la hoja de orden, e insertarla en la bandeja de órdenes confirmadas	No
8100	Null	Valor a ignorar	Sí

Cuadro 4.1: Tabla de operaciones

4.3.2. Acciones

ID	Operación	Acción	ignorar
101	Recoger	Recoger hoja.	No
103	Recoger	Recoger objeto de la caja.	No
107	Recoger	Recoger hoja de caja.	No
108	Recoger	Caminar hacia la mesa de trabajo	No
109	Recoger	Poner la caja empaquetada	No
201	Trasladar etiqueta del objeto	Quitar la etiqueta del objeto	No
202	Trasladar etiqueta del objeto	Poner la hoja de orden	No
203	Trasladar etiqueta del objeto	Sujetar bolígrafo	No
204	Trasladar etiqueta del objeto	Escribir marca de comprobación	No

205	Trasladar etiqueta del objeto	Poner el objeto en la bolsa	No
301	Ensamblar caja	Coger el cartón	No
302	Ensamblar caja	Doblar la tapa	No
303	Ensamblar caja	Cerrar el fondo	No
304	Ensamblar caja	Girar la caja	No
305	Ensamblar caja	Coger caja ensamblada	No
401	Insertar objeto	Insertar objeto en la caja.	No
402	Insertar objeto	Hinchar almohadillas.	No
403	Insertar objeto	Separar almohadillas	No
404	Insertar objeto	Poner objeto en la bolsa pequeña	No
501	Cerrar caja	Mover el paquete completo a la mesa trasera	No
502	Cerrar caja	Poner el objeto en la bolsa	No
601	Pegar etiqueta a la caja	Coger el cartón	No
701	Escanear etiqueta	Doblar la tapa	No
702	Escanear etiqueta	Cerrar el fondo	No
703	Escanear etiqueta	Girar la caja	No
704	Escanear etiqueta	Coger caja ensamblada	No
705	Escanear etiqueta	Insertar objeto en la caja.	No
706	Escanear etiqueta	Hinchar almohadillas.	No
707	Escanear etiqueta	Separar almohadillas	No
801	Pegar etiqueta de envío	Coger la etiqueta de envío	No
802	Pegar etiqueta de envío	Pegar etiqueta de envío	No
901	Colocar en la mesa trasera	Coger caja ensamblada	No
902	Colocar en la mesa trasera	Poner caja ensamblada.	No
1001	Cumplimentar la orden	Coger el rotulador.	No
1002	Cumplimentar la orden	Escribir firma.	No
1003	Cumplimentar la orden	Poner la orden en la bandeja.	No
8101	Null	Otros	Sí
8102	Null	Error del sistema	Sí

8103	Null	Ignorar	Sí
8104	Null	Desconocido	Sí
8201	Error	Error.	Sí

Cuadro 4.3: Tabla de acciones

4.3.3. Ejemplo

La manera de asignar estas anotaciones a las operaciones realizadas por los usuarios es más complicada que en los conjuntos de datos utilizados habitualmente. En este caso no se dispone de un fichero en el que estén, linea a línea, las acciones realizadas por el usuario y sus anotaciones, sino que están distribuidas entre varios ficheros. Los datos están separados, en primera instancia, por usuario, teniendo cada usuario una carpeta separada. Dentro de cada usuario, cada sesión de pruebas tiene también una carpeta, y dentro de cada sesión hay tantas carpetas como sensores distintos se hayan utilizado en esa sesión. Como se verá en el resto de este apartado, es en estas carpetas de los sensores donde se almacenan las mediciones que realizan, además del tiempo en el que se realizaron. El valor de tiempo es el que se usa para cruzar esas tablas con el fichero de anotaciones, y hacer corresponder a cada medición su acción u operación correspondiente. Como ejemplo, se muestra la lectura de la primera sesión del usuario s103: Cuando se lean las mediciones de los sensores, se utilizará este campo *unixtime* para obtener la

	unixtime	user	session	box	operation	action
0	1636688346000	U0103	S0100	11	100	101
1	1636688347000	U0103	S0100	11	100	101
2	1636688348000	U0103	S0100	11	100	101
3	1636688349000	U0103	S0100	11	100	101
4	1636688350000	U0103	S0100	11	100	101
...	...	...	...	...	...	...
1189	1636689535000	U0103	S0100	20	900	8100
1190	1636689536000	U0103	S0100	20	900	8100
1191	1636689537000	U0103	S0100	20	1000	1001
1192	1636689538000	U0103	S0100	20	1000	1002
1193	1636689539000	U0103	S0100	20	1000	1002

1194 rows × 6 columns

Figura 4.1: Sesión s100 del usuario s103

etiqueta correspondiente. También podemos ver en la figura inferior cómo evolucionan las clases a las que pertenecen las actividades a lo largo del tiempo dentro de una sesión. Hay un patrón constante, ya que las acciones, si se realizan correctamente, deben seguir una secuencia estándar:

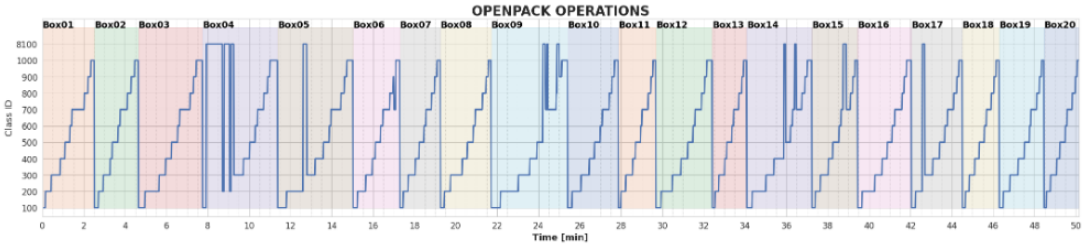

Figura 4.2: Evolución de las actividades durante la ejecución de una sesión de trabajo.

4.4. Fuentes de datos

Para poder recabar la información, todos los usuarios portaban, o eran monitorizados, por tres tipos de dispositivos diferentes: **Kinect**, **ATR-TSND151** y **Empatica E4**. A su vez, cada uno de estos dispositivos da lugar a distintas fuentes de datos. Los datos que se obtienen de cada uno de los sensores están separados por usuario, y por sesión del usuario, y consisten en un *csv* en el que una de las columnas contiene el tiempo en el cual se realizó la medición (en *unixtime*, necesario para cruzar datos con las tablas de anotaciones), y el resto es la información en bruto proporcionada por el sensor. Se detallan uno a uno los diferentes dispositivos a continuación:

4.4.1. ATR-TSND151

El ATR-TSND151 es una unidad de medición inercial (en inglés **IMU**, *Inertial measurement unit*), un dispositivo que, acoplado a un cuerpo, puede medir variables como la velocidad, la aceleración o la orientación. El dispositivo utilizado para el desafío reporta información acerca de la aceleración, el giroscopio, y la orientación (como cuaterniones). Cada uno de los usuarios porta cuatro de ellos, con lo cuál es necesario cruzar los datos para obtener las 4 mediciones en un mismo dataframe (la función *load_imu()*, incluida en las librerías proporcionadas por la organización, se encarga de forma transparente).

Figura 4.3: Dispositivo ATR-TSND151. [1]:

Cada uno de los dispositivos aporta la siguiente información:

1. Acc_x, Acc_y, Acc_z: valores de la aceleración en cada uno de los ejes de coordenadas
2. Gyro_x, Gyro_y, Gyro_z: valores del giroscopio en cada uno de los ejes de coordenadas.
3. Quat_w, Quat_x, Quat_y, Quat_z: cuaternión en cuatro ejes de coordenadas.

[1]Fuente: `https://www.atr-p.com/products/sensor.html`

Además, cada uno de los usuarios lleva cuatro dispositivos distintos, situados en cuatro lugares del cuerpo diferentes:

1. Atr01: Muñeca derecha
2. Atr02; Muñeca izquierda
3. Atr03: Brazo derecho
4. Atr04: Brazo izquierdo.

	unixtime	acc_x	acc_y	acc_z	gyro_x	gyro_y	gyro_z	quat_w	quat_x	quat_y	quat_z
0	1636688344032	0.7869	0.6212	0.1482	1.16	-1.67	0.83	0.0	0.0	0.0	0.0
1	1636688344065	0.7852	0.6193	0.1390	1.24	-1.18	0.60	0.0	0.0	0.0	0.0
2	1636688344098	0.7887	0.6222	0.1358	0.95	-0.14	0.49	0.0	0.0	0.0	0.0
3	1636688344131	0.7916	0.6198	0.1219	1.04	-0.65	0.28	0.0	0.0	0.0	0.0
4	1636688344164	0.7816	0.6232	0.1163	1.16	0.62	0.45	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...
36358	1636689543846	0.0714	0.6730	1.1490	43.59	-82.92	-138.43	0.0	0.0	0.0	0.0
36359	1636689543879	0.9586	0.7138	0.7933	28.14	-20.02	-110.70	0.0	0.0	0.0	0.0
36360	1636689543912	0.1681	0.9066	0.7261	23.35	20.96	-105.35	0.0	0.0	0.0	0.0
36361	1636689543945	0.3861	0.9625	0.4942	-4.86	0.36	-56.29	0.0	0.0	0.0	0.0
36362	1636689543978	0.2220	0.9537	0.6824	-42.55	-2.13	7.47	0.0	0.0	0.0	0.0

Figura 4.4: Ejemplo de la información aportada por uno de los dispositivos ATR-TSND151.

El ejemplo anterior se obtuvo de sólo uno de los dispositivos, en concreto del situado en el brazo derecho. Para el entrenamiento del modelo se usarán los cuatro dispositivos que porta cada sujeto, con lo que se tendrá cuatro veces una información similar a la mostrada en la imagen previa. Las mediciones se realizan a una frecuencia de 30 Hz, lo cual es importante para sincronizar los datos de las distintas modalidades, como se explicará en la sección de desarrollo del modelo.

En la figura 4.5 se muestra una distribución de ejemplo de cómo son los datos que nos devuelve este sensor. Se ha elegido el segundo dispositivo, el usuario tercero, y la sesión segunda, pero cualquier combinación de estos elementos mostraría una distribución similar. En la primera parte de la gráfica, se muestra la evolución temporal de los valores que toma el dispositivo a lo largo de la ejecución de una sesión. Están agrupados por la modalidad medida (aceleración, giroscopio, cuaternión), y, en cada tipo, se encuentran superpuestos con diferentes colores cada una de las distintas mediciones posibles en cada modalidad. La parte inferior de la gráfica está estructurada de manera similar, pero en este caso se muestra la distribución de valores que toma cada de las modalidades. Como se puede ver, cuaternión toma siempre el valor 0 en todas sus mediciones posibles: es un sensor que no funciona correctamente en algunos dispositivos, y por tanto se omitirá al entrenar el modelo.

4.4.2. Kinect

El Microsoft Kinect fue un dispositivo de entrada que detectaba movimientos desarrollado por Microsoft para sus consolas de juegos Xbox y PCs con Windows. Fue presentado por primera vez en noviembre de 2010 para la consola Xbox 360. El Kinect utilizaba una combinación de cámaras, sensores de profundidad y micrófonos para rastrear e interpretar los movimientos y comandos de voz de los usuarios. Originalmente se diseñó para permitir una integración más

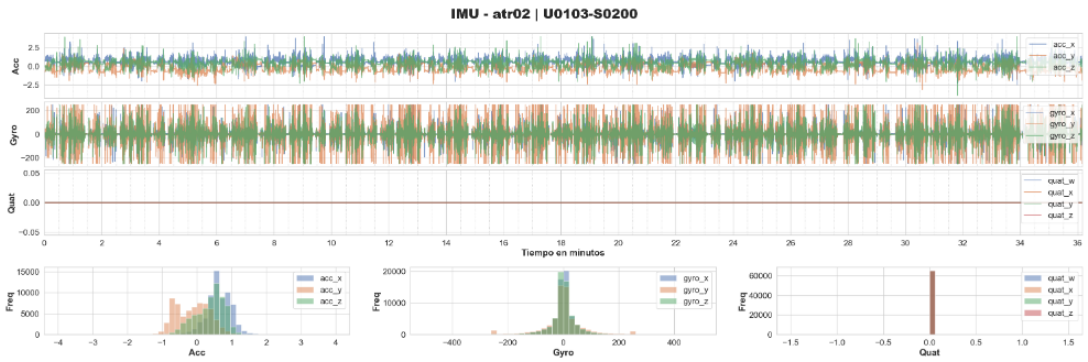

Figura 4.5: Distribución de los datos aportados por el segundo sensor, para el usuario 103, y la sesión 200

natural con juegos y aplicaciones, pero encontró aplicaciones en los campos de la medicina, la educuación y la realidad virtual. En el conjunto de datos proporcionado, el sensor Microsoft

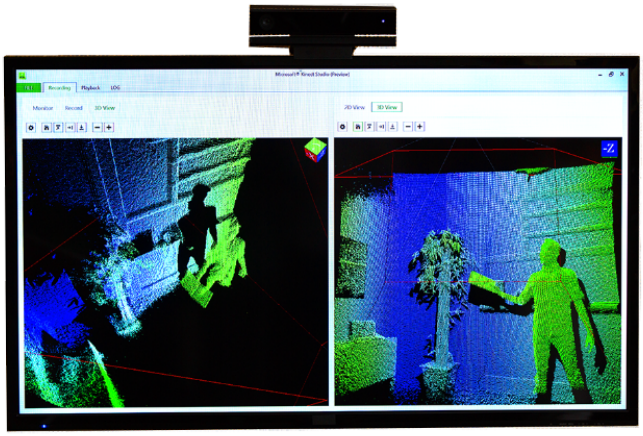

Figura 4.6: Ejemplo de vídeo capturado por el sensor Kinect. [2]

Kinect se utiliza para extraer los «puntos clave¿"(*keypoints*) del usuario. En el campo del procesamiento de imágenes y de visión por computador, se entiende por *keypoints* puntos específicos e identificables en una imagen o conjunto de datos que tienen alguna relevancia en el contexto del análisis o procesamiento que se está realizando. Estos puntos son seleccionados debido a su importancia para extraer información o características específicas. En este caso, los puntos claves son las diversas partes del cuerpo del usuario, de las que se mide su situación espacial.

[2]Fuente: `https://learn.microsoft.com/en-us/windows/apps/design/devices/kinect-for-windows`

Nª	ID	Localización
0	0	Nariz
1	1	Ojo izquierdo
2	2	Ojo derecho
3	3	Oreja izquierda
4	4	oreja derecha
5	5	Hombro izquierdo
6	6	Hombro derecho
7	7	Codo izquierdo
8	8	Codo derecho
9	9	Muñeca izquierda
10	10	Muñeca derecha
11	11	Cadera izquierda
12	12	Cadera derecha
13	13	Rodilla izquierda
14	14	Rodilla derecha
15	15	Tobillo izquierdo
16	16	Tobillo derecho

Cuadro 4.4: Tabla con el listado de puntos clave cuyas coordenadas devuelve el sensor

Para llevar a cabo esta medición, el dispositivo graba al usuario realizando la tarea. Posteriormente, partiendo de la grabación, se extrajeron los puntos utilzando **MMpose** [27], y se almacenaron en el conjunto de datos. Son estos puntos, y no directamente la imagen, o el vídeo, lo que contienen los datos, por respeto a la privacidad de los usuarios que participaron en las pruebas. Los datos no están guardados como un *csv*, sino utilizando el formato MS COCO JSON. Leer las mediciones de Kinect será leer una lista de archivos en este formato, cada uno asociado a una sesión de pruebas. El objeto raíz proporciona metadatos que incluyen, entre otros, la fecha de creación, la sesión a la que pertenecen las mediciones que contiene o la licencia de uso. Dentro de cada uno de estos objetos, existe una lista de diccionarios llamada *annotations*, Cada una de las entradas de esta lista se corresponde con una medición realizada por el dispositivo, y contiene, entre otros metadatos, el momento en UNIXtime en el que se tomó la medición, y las coordenadas de cada uno de los puntos clave. La tabla 4.5 resume este formato:

Nº	Nombre	Descripción	Unidad	Tipo
0	info		None	Dict
1	Licenses		None	list[Dict]
2	Annotations		None	int
3	Annotations/id		None	int
4	Annotations/image_id	Valor de UNIXtime en milisegundos de cuando se realizó la medición	Milisegu	int
5	Annotations/ category_id		None	int
6	Annotations/ area		None	float
7	Annotations/ bbox	[x, y, ancho, alto]	None	Tuple[float, float, float, float]
8	Annotations/ iscrowd		None	int
9	Annotations/ keypoints	[x1, y1, v1, ...] (v? = puntuación de confianza)	píxel	list[float]
10	Annotations/ num_keypoints		None	int
11	Annotations/ bbox_score	Puntuación de la confianza acerca de los límites de la caja	None	float
12	Annotations/ track_id		None	int
13	Category		None	list

Cuadro 4.5: Tabla con el formato de los archivos con los datos proporcionados por Kinect

A modo de ejemplo, el campo *info* del objeto de la segunda sesión del tercer usuario:

```
{'year': 2022,
 'version': 'v0.2.0',
 'description': 'openpack/U0103/2d-kpt/mmpose-hrnet-w48-
 posetrack18-384x288-posewarper-stage2/single/S0200',
 'contributor': 'Naoya Yoshimura, Jaime Morales, Takuya Maekawa',
 'url': 'https://open-pack.github.io',
 'date_created': '2022/03/31'}
```

Y la información acerca de los puntos clave, contenida en el campo *annotations*:

```
 {'id': 1634869192038000,
 'image_id': 1634869192038,
```

```
'category_id': 1,
'area': 56387.824,
'bbox': [597.3, 320.4, 174.6, 322.9],
'iscrowd': 0,
'keypoints': [[698.8, 373.7, 0.92614394],
 [688.2, 407.5, 0.83524674],
 [690.3, 350.5, 0.8222388],
 [717.8, 380.0, 0.29044425],
 [658.7, 363.2, 0.42727488],
 [738.8, 426.4, 0.86197984],
 [635.5, 424.3, 0.85390973],
 [753.6, 521.3, 0.7679441],
 [618.6, 506.6, 0.8285348],
 [757.8, 597.3, 0.88464993],
 [610.2, 586.7, 0.8356974],
 [711.4, 586.7, 0.6472243],
 [641.8, 582.5, 0.6692864],
 [696.7, 673.2, 0.43312368],
 [641.8, 664.7, 0.42528123],
 [724.1, 628.9, 0.16155383],
 [622.9, 626.8, 0.10479919]],
'num_keypoints': 17,
'bbox_score': 0.9971596,
'track_id': 106}
```


Cada una de las 17 entradas del array corresponde, por orden, a las partes del cuerpo enumeradas previamente.

4.4.3. E4

Además de todo lo anterior, los usuarios también portaban dos pulseras *empatica E4*. Cada uno de los sujetos lleva dos de estos dispositivos, uno en cada muñeca, que proporcionan información acerca de la aceleración, el pulso, la temperatura y la actividad electrotermal. Los dos sensores están numerados como *e401* (muñeca derecha) y *e402* (muñeca izquierda). Se muestra una lectura de ejemplo de los datos de aceleración del dispositivo *e401*, correspondiente a la segunda sesión del tercer usuario:

También, en la figura 4.8 se muestra la secuencia de valores que toman cada una de las modalidades posibles a lo largo del tiempo, y su distribución de datos, para el mismo sensor, usuario y sesión de la lectura de aceleración previa.

4.5. Usuarios

Para garantizar una mayor generalidad en los datos, participaron 16 sujetos distintos de entre 20 y 50 años de edad en la realización de las tareas, de los que se obtuvieron los datos utilizando los sensores anteriormente descritos. Además, cada uno de estos sujetos desempeñó las operaciones que se quieren clasificar a lo largo de varias sesiones, para tomar en cuenta que

	time	acc_x	acc_y	acc_z
0	1636693887007	-0.022306	-0.030273	0.092412
1	1636693887038	-0.017526	-0.030273	0.094005
2	1636693887070	-0.017526	-0.031866	0.092412
3	1636693887101	-0.022306	-0.033459	0.092412
4	1636693887132	-0.025493	-0.031866	0.092412
...	...	...	...	...
68667	1636696032874	-0.073292	-0.022306	0.066919
68668	1636696032905	-0.065326	-0.020713	0.074885
68669	1636696032937	-0.058952	-0.015933	0.073292
68670	1636696032968	-0.071699	-0.017526	0.068512
68671	1636696032999	-0.063732	-0.014340	0.073292

Figura 4.7: Lectura de aceleración del sensor e4

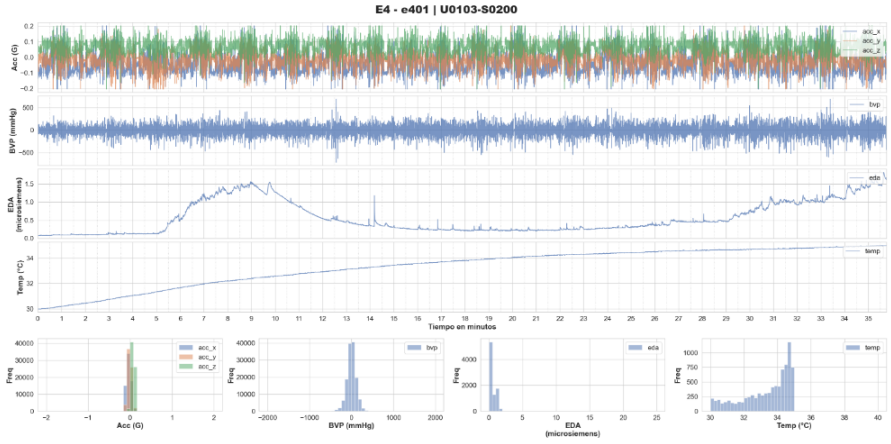

Figura 4.8: Distribución de datos del sensor e4

no siempre el trabajo se desempeña de la misma manera: a veces se está más cansado, se tiene más prisa por terminar, los objetos a empacar pueden ser de distintos tamaños e implicar por tanto unos gestos diferentes (que se traducirán en mediciones distintas), etc...

La tabla 4.6 resume la información de los sujetos que participaron en el proyecto:

Sujeto	Sexo	Edad	Mano dominante	Experiencia	Nota
U0101	M	-	Derecha	-	-
U0102	M	-	Derecha	-	-
U0103	M	50s	Derecha	6 meses	-
U0104	M	50s	Derecha	1 mes	-
U0105	M	30s	Derecha	4 años	-
U0106	M	40s	Derecha	1 mes	-
U0107	M	40s	Derecha	3 años	-
U0108	M	30s	Derecha	3 años	-
U0109	H	30s	Izquierda	6 meses	-
U0110	M	40s	Derecha	10 meses	-
U0111	M	50s	Derecha	2 años	-
U0202	M	30s	Derecha	4 años	Misma persona que U0105
U0203	M	30s	Derecha	3 años	Misma persona que U0108
U0204	M	40s	Derecha	1 año	Misma persona que U0110
U0205	M	40s	Derecha	3 años	Misma persona que U0107
U0207	M	40s	Derecha	20 meses	-
U0210	M	50s	Derecha	3 meses	Misma persona que U0103

Cuadro 4.6: Tabla con información personal de los sujetos que participaron en las pruebas

Además, para aumentar la diversidad de los datos, se asignó a cada sesión un tipo de escenario. Cada escenario consiste serie de instrucciones específicas que los sujetos debían seguir, de modo que todas las operaciones dentro de esa sesión estuvieran sesgadas de la manera indicada. Las consignas fueron:

1. **Escenario 1:** Los sujetos que siguieran las instrucciones de trabajo en la medida de lo posible. La lista de artículos de un pedido se determinó a partir de hojas de pedido recogidas en centros logísticos reales, y las probabilidades con las que se incluían artículos muy grandes o pequeños en un pedido eran mayores que las probabilidades reales.

2. **Escenario 2:** Los sujetos a alterar el procedimiento de las operaciones a su discreción. Además, las probabilidades con las que se incluían artículos bastante grandes o pequeños en un pedido eran menores que las del escenario 1 (similares a las probabilidades reales),

3. **Escenario 3:** Se añadieron situaciones/acciones irregulares al escenario 2. (i) Algunas cajas de envío ya estaban montadas (debido a errores, por ejemplo) y estaban a disposición de los trabajadores para su uso. (ii) Cuando en un pedido se incluían artículos pequeños (como pilas secas), también se pedía al sujeto que los metiera en una bolsa de papel. (iii) Cuando varios pedidos consecutivos incluían sólo artículos pequeños, un sujeto podía llevarlos de la mesa de atrás al banco de trabajo al mismo tiempo.

4. **Escenario 4:** Se apresuró a los sujetos introduciendo una alarma auditiva en el Escenario 3. Medimos la duración media de un periodo de trabajo realizado por un sujeto en sesiones anteriores, y se iniciaron alarmas periódicas (intervalo de 30-45 s) cuando el tiempo transcurrido de un periodo superaba aproximadamente el 80 % de la duración media.

Capítulo 5

Comparación de distintas estrategias de fusión

En esta sección se comparan las tres estrategias de fusión posibles. El modelo (CSNet) ha de ser único en las tres, para que las diferencias que se den entre los distintos tipos sean representativas, y debidas únicamente al tipo de fusión escogida. Las tres modalidades que se van a fusionar son el sensor IMU, los puntos clave que proporciona Kinect, y el sensor E4:

Sensor	Tipo de lecturas	Dimensión de lectura	Frecuencia de lectura
IMU	Acelerómetro en los tres ejes de coordenadas	4 sensores por cada sujeto: 12 dimensiones	30 Hz (1800 lecturas por minuto)
Puntos clave de Kinect:	17 puntos clave, izquierda y derecha	34	15 Hz (900 lecturas por minuto)
E4	Acelerómetro en los tres ejes de coordenadas	Dos pulseras por sujeto: 6 dimensiones	32 Hz (1920 lecturas por minuto)

Cuadro 5.1: Tabla con información de los tipos de sensores empleados en la clasificación

Aunque en el conjunto de datos hay algunas más disponibles, se ha restringido la clasificación a estas variables, porque las otras modalidades contienes alguna medida errónea que puede perjudicar el proceso de aprendizaje. Trabajar con sensores con distintas tasas de muestreo es una tarea compleja, que en cada una de las técnicas se resuelve de forma distinta.

5.1. Modelo CSNET

El modelo que se ha utilizado para este objetivo es el CSNet, propuesto en [13]. CSNet es una arquitectura que combina redes convolucionales con mecanismos de atención. Las redes convolucionales detectan características a nivel local, y posteriormente a esas características les son asignados pesos que posibilitan la detección de relaciones temporales a largo plazo.

En la figura 5.1 se muestra la composición del modelo. Los datos de entradas atraviesan tres bloques: convolución, auto-atención y convolución. Los bloques de convolución están compuestos por dos unidades similares consecutivas, que aplican: 1) convolución de 1 dimensión, normalización por lotes, y ReLu como función de activación, 2) pasan de nuevo por el mismo

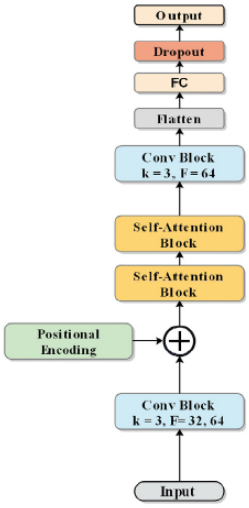

Figura 5.1: Esquema de la arquitectura CSNet. Fuente original [13]

proceso, y 3) finalmente son agregadas para reducir la resolución de los datos. Uno de los problemas de la auto-atención es que procesan toda la secuencia al mismo tiempo, sin tener en cuenta las relaciones temporales entre unos elementos y otros, por tanto es necesario añadir una codificación de posicional que permita distinguir qué elemento va antes, y cuál después. La siguiente etapa son los dos bloques de auto-atención. Cada bloque tiene dos partes: la auto-atención, y la red *feed-forward*, que se encarga de aprender relaciones entre los datos de entrada. El siguiente paso es volver a aplicar un nuevo bloque convolucional a los datos, y por último una capa totalmente conectada se encarga de llevar a cabo la clasificación final. En la figura 5.2 se muestra la arquitectura interna de los bloques.

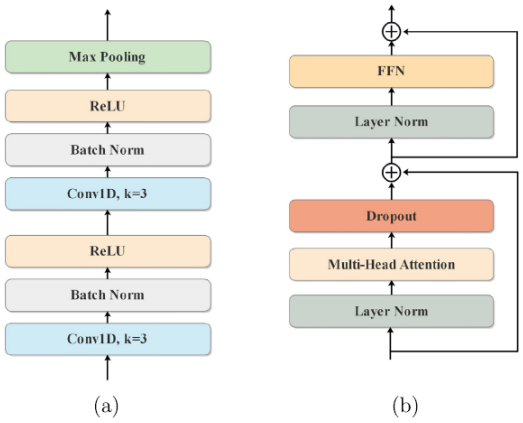

Figura 5.2: Estructura de los bloques de a) convolución, b) auto-atención . Fuente original [13]

La implementación del modelo se realizó en *Pytorch*. Cada uno de los bloques es una clase distinta, que hereda de la clase *Module*, propia de Pytorch, pensada para que sea la clase base de todos los modelos que se desarrollen. El modelo completo (previo a cualquier fusión de datos) se implementó en la clase *CSNetBlock*. El bloque de convolución se implementó con la clase *ConvolutionalBlock*, y el de atención con *SelfAttentionBlock*. La diferencia es que, en lugar de añadir una capa completamente conectada, la última etapa consiste en un bloque que

redimensiona los datos de salida, *ReshapeBlock*. Este último bloque es necesario para poder efectuar la fusión de características, como se explica en la sección correspondiente. La salida del modelo será un tensor de 11 canales (uno por cada clase posible, más la clase "no válido"), y de la longitud que tenga la secuencia. Hay que tener en cuenta que a cada elemento de la secuencia de salida le corresponde un instante de tiempo. Ya que se sabe, por cada usuario, qué actividad estaba realizando a cada momento, el último paso es aplicar soft-max para obtener la predicción de la clase más probable para cada elemento, y comparar con la actividad que constituye el *ground truth* para computar la pérdida.

La justificación del modelo elegido se basa en que:

1. En el paper, los autores detallan de manera clara y concisa la arquitectura, lo que simplifica el proceso de codificación.

2. La arquitectura modular propuesta permite hacer fusión de características en lugares obvios.

3. Obtiene unos buenos resultados en los *benchmarks* habituales para estas tareas.

4. La mayoría de los modelos propuestos en la literatura utilizan RNN (usualmente LSTM), pero no este modelo, lo que es interesante para comparar la efectividad de unos y otros.

Un último detalle importante es que todos los modelos desarrollados en los trabajos que se han reseñado terminan con una capa completamente conectada. Esta es la capa que lleva a cabo la clasificación, con tantas unidades como clases distintas tenga el proyecto. En este caso, es necesario cambiar la arquitectura, sustituyendo la capa completamente conectada por una capa convolucional de 1 dimensión, que será la que realice la clasificación. La idea de sustituir la capa completamente conectada por una capa convolucional se encuentra en [35], donde se detalla su uso en la segmentación de imágenes. Una capa completamente conectada es un caso específico de una capa convolucional, en la que existen tantos filtros como unidades de salida, y el tamaño del kernel ocupa la entrada completa, Se puede realizar una por tanto sustitución, con la ventaja de aprovechar la superposición en la red convolucional para disminuir el número de conexiones totales. El formato de salida de la competición, que aunque no se vaya a participar se ha respetado para poder hacer uso de las librerías que computen la puntuación de los modelos desarrollados, implica que el problema se asemeja conceptualmente a uno de segmentación semántica: no hay que devolver la clase a la que pertenece una secuencia, si no la clase a la que pertenecen cada uno de los elementos de la secuencia. Esto impide que se utilicen redes completamente conectadas en la salida ya que, por ejemplo, dado que hay 11 tipos distintos de acciones a clasificar, y uno secuencia de 1 minuto contiene (en el caso del sensor IMU) 1800 elementos, solamente la última capa serían 19800 conexiones, multiplicados por la dimensión de la entrada. Es decir, que habría, como mínimo, varios millones de parámetros entrenables tan sólo en la última capa. El uso de redes completamente convolucionales permite solucionar este problema: se utilizan tantos filtros como clases haya, cada uno de ellos produciendo un «mapa de calor» con la puntuación que otorga a cada elemento en la secuencia de pertenecer a esa clase.

5.2. Fusión a nivel de sensor

Siguiendo a [23], la fusión de sensor consiste en obtener las mediciones de cada uno de los sensores, y unirlas en un sólo tensor antes de proceder a la clasificación. Sumando las dimensiones

de todas las modalidades de entrada, como se indicó en la tabla 5.1, el tensor resultante tiene 52 canales de entrada a la primera convolución, y tantos elementos como longitud tenga la secuencia (sobre esto, más adelante).

5.2.1. Preprocesamiento de datos

El hecho de que diferentes sensores muestreen a distintas frecuencias supone que la cantidad de muestras que se obtienen en una ventana de tiempo es diferente para cada uno. El preprocesamiento previo de los datos tiene que tener en cuenta esto al construir el conjunto de datos con el que se va a efectuar el entrenamiento (o los tests). El problema el siguiente: Se tienen una serie de mediciones de varios sensores que abarcan grandes intervalos de tiempo, y separadas por varios sujetos. Ya que el problema consiste en la clasificación elemento a elemento, se tiene que elegir una ventana de tiempo para crear secuencias más pequeñas del tamaño de la ventana dentro de todas las modalidades. El modelo se ha entrenado utilizando como ventana de tiempo 1 minuto. Como las tasas de muestreo son distintas para cada sensor, en un minuto dado cada uno de ellos realiza un número de lecturas distintas.

Una posible solución sería utilizar el sensor que menor tasa de muestreo tenga, y por cada una de sus lecturas asignar la lectura de los otros sensores que más próxima en el tiempo sea. El problema de esta aproximación es que, según Khan et al. [22], para que el reconocimiento de actividades humanas no se vea afectado, es necesario una tasa mínima de muestreo, por debajo de la cual se empieza a perder definición. El valor de esta tasa de muestreo depende de la actividad a clasificar y el sensor, pero en todos los casos los autores determinaron que esta por encima de 20 Hz, de otro modo la distribución de datos muestreados comienza a separarse por encima del 5 % de la distribución original. Dado que el sensor Kinect muestrea por debajo de ese umbral, a 15 Hz, se ha optado por otra solución: utilizar los datos del sensor IMU como referencia, y asignar a cada lectura la de los otros sensores que sea más cercana en el tiempo. Una vez se ha hecho un *merge* de todas lecturas de este modo, interpolar los valores ausentes entre los datos, que corresponden con los huecos que han ido quedando por tener menor frecuencia de muestreo. Las figuras 5.3 y 5.4 muestran esta operación con datos ilustrativos como ejemplo:

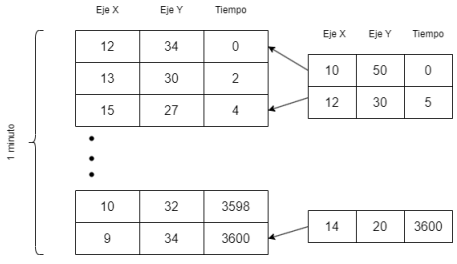

Figura 5.3: Asignación de los elementos de la secuencia

Esta técnica, que es simplemente una forma de imputación de datos ausentes, tiene el problema de que añade información que no existe, pero la distribución de los datos es más similar a la original que si se duplicasen los valores en las entradas a las que le falten datos, y no se pierde la información que otorgan los sensores tasa de muestreo más alta.

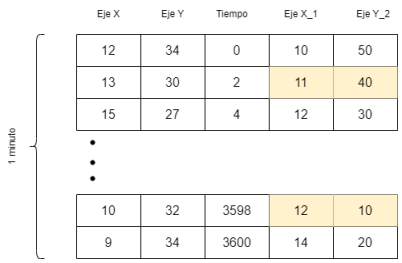

Figura 5.4: Interpolación

5.2.2. Evaluación

En las figura A.1 y A.2 se muestra la evolución del entrenamiento en los conjuntos de entrenamiento y validación. Aunque el modelo logra aprender los datos, se puede apreciar en el conjunto de validación que no logra converger. Los resultados en el conjunto de test son también negativos:

name	precision	recall	f1
avg/macro	0.334235	0.231302	0.217394
avg/weighted	0.371666	0.262573	0.244682
Picking	0.229927	0.808635	0.358047
Relocate Item Label	0.226433	0.369112	0.280681
Assemble Box	0.452525	0.175686	0.253107
Insert Items	0.000000	0.000000	0.000000
Close Box	0.416149	0.075281	0.127498
Attach Box Label	0.447514	0.225000	0.299445
Scan Label	0.938889	0.327202	0.485284
Attach Shipping Label	0.464789	0.113597	0.182573
Put on Back Table	0.010736	0.025362	0.015086
Fill out Order	0.155388	0.193146	0.172222

Figura 5.5: Resultados del modelo con fusión a nivel sensor en el conjunto de test

En la figura 5.5 pueden verse la precisión, la cobertura y la puntuación f1, de cada una de las clases que forman el conjunto de datos y en total. Este será el formato que se va a utilizar de ahora en adelante para comparar unos resultados con otros. Como se puede ver, los resultados no han sido buenos, alcanzándose sólo un 0.2446 de puntuación f1 ponderada por clases.

5.3. Fusión a nivel de características

Para realizar la fusión a nivel de características, es necesario cambiar la manera en que se construye el conjunto de entrenamiento. En la opción anterior se hacía un *merge* de las distintas modalidades antes de la clasificación. Ahora el *merge* se va a hacer después, una vez se hayan obtenido las características, pero para eso antes hay que configurar los datos, que tienen que entrar sincronizados al modelo. El algoritmo del proceso es:

1. Una ventana de tiempo va a ser clasificada. Por las diferentes tasas de muestreo, la misma ventana tiene diferente número de muestras dependiendo del sensor.

2. La ventana se separa en tantas secuencias como modalidades distintas existan.

3. Cada una de las modalidades atraviesa su propio bloque CSNet, que aprende características de alto nivel.

4. Como el número de muestras era distinto en cada modalidad, los tensores de características tienen tamaños distintos, por tanto se tienen que redimensionar a un tamaño homogéneo.

5. Una vez que los tensores de características de todas las modalidades miden lo mismo, se concatenan.

6. Una última capa produce la salida con las clasificaciones.

En la figura 5.6 se muestra un diagrama del proceso. A cada ventana de tiempo de 1 minuto le corresponden una serie de muestras (1800 para IMU, 900 para los puntos clave de Kinect, 1920 para E4), cada muestra teniendo un tamaño específico según la información que mide cada sensor. Por separado, tres bloques CSNet extraen las características de cada uno. Las dimensiones de la salida son de 64 canales por la mitad de la longitud de la secuencia de entrada (debido al *max-pooling*). Antes de ser concatenadas, se redimensionan a 64 x 1800 para que tengan un tamaño homogéneo. La justificación es que los datos de IMU tienen menos errores que los otros dos sensores, por lo que para clasificar en el último paso da menos problemas ajustarlo a su tamaño. Los tres tensores de características del mismo tamaño se concatenan, y se procede a aplicar una última capa de convolución que devuelve un tensor de 1800 por 11 (la longitud de la secuencia por el número de clases), que posteriormente pasa a una capa *soft-max* a partir de la cual se calcula la pérdida.

5.3.1. Preprocesamiento de datos

El preprocesamiento de los datos en este caso va a consistir en generar las ventanas de tiempo para que los tres sensores estén sincronizados. Es decir, cada uno de los elementos de un *batch* durante el entrenamiento está formado por una secuencia que proviene del sensor IMU, una de Kinect, y una de E4. Para poder gestionar los datos de esta manera, el proceso fue el siguiente:

1. El conjunto de datos está formado por una serie de usuarios, a cada uno de los cuales les corresponde una serie de sesiones.

2. Los datos se leen al completo y en secuencia, es decir, en un bucle anidado en la que para cada usuario se leen todas sus sesiones. El objetivo será «aplanar» los datos, para que todas las mediciones formen una lista.

3. La gestión de los diferentes sensores se lleva a cabo mediante un diccionario, con tres claves: *imu*, *keypoints* y *e4*. Cada una de ellas contiene una lista de las mediciones de cada sensor.

4. Las tres listas tienen que tener la misma cantidad de elementos. Esto es obligatorio por como funciona el entrenamiento: para formar un *batch* se toma un elemento al azar de cada una de las tres listas al mismo tiempo, se concatenan, y es a la entrada del modelo dónde se separan por modalidad, para ser luego unidas sus características.

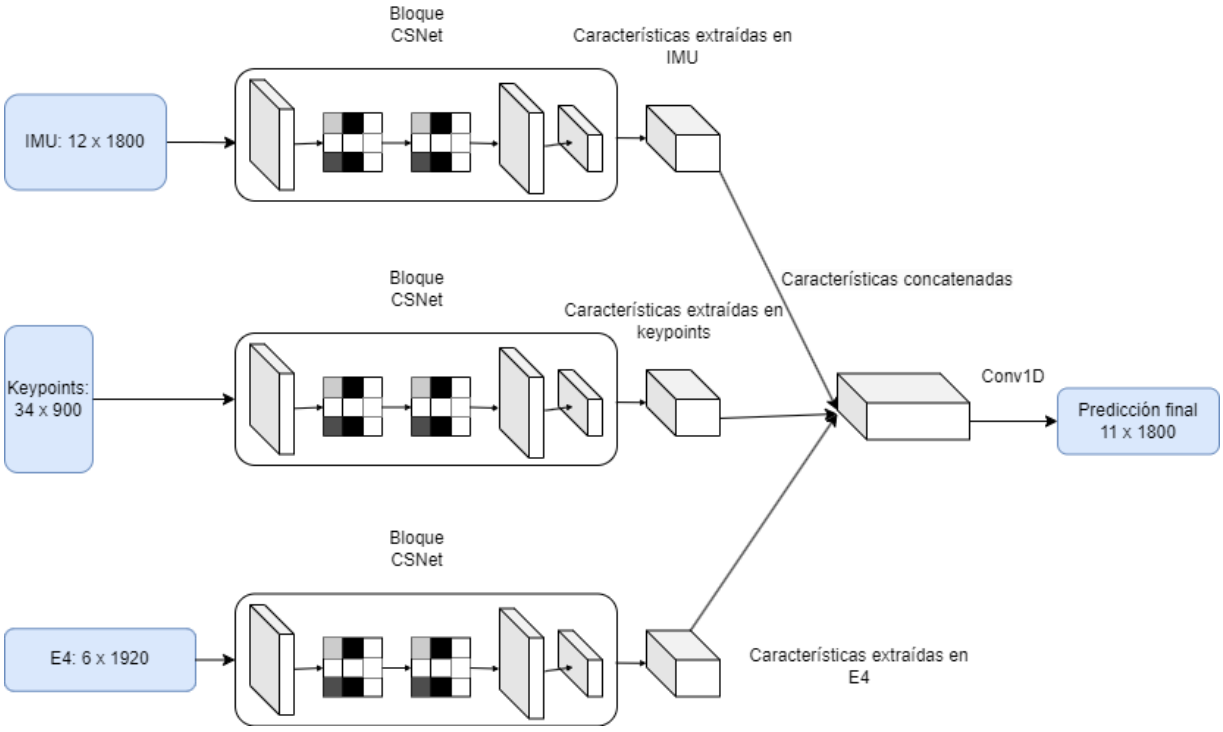

Figura 5.6: Esquema de la fusión de características.

5. Sucede con frecuencia que un sensor se quede a mitad de una ventana de tiempo. Es decir que para un minuto dado, uno de los sensores sí que tenga lecturas completas, pero algún otro haya dejado de leer en un punto intermedio. En este caso, se procede a rellenar con 0 el vector de lecturas, y se etiquetan con la clase nula, que implica que esas lecturas no computan para la función de pérdida.

En la figura 5.7 se muestra un diagrama de ejemplo de un caso en el que, en una ventana de tiempo concreta, el sensor IMU sí tuviese lecturas suficientes, pero los otros dos no, mostrando en fondo amarillo las celdas que corresponden a los instantes de tiempo que tuvieron que ser completados.

5.3.2. Evaluación

A pesar de que en esta sección no se están afinando hiperparámetros, para centrarse únicamente en la fusión, hay una de las características de la red que sí influye directamente en cómo se combinan las características: la dimensión de la capa de codificación. Cuando los bloques CSNet ya han extraído las características, la capa que aplana los tensores resultantes puede tener distinto número de unidades. Un número mayor de unidades mejora la precisión, pero el número de parámetros entrenables crece exponencialmente (al ser una capa completamente conectada), lo que obliga a buscar una solución intermedia. Se reportan los resultados con 60 y 84 unidades como ejemplo, a partir de ese valor el número de parámetros entrenables empezaba a volverse inmanejable, y la precisión no aumentaba de forma sustancial.

En las figuras A.3 y A.4 se muestra la evolución del entrenamiento del modelo usando 64 unidades en la capa de codificación. La curva en el conjunto de entrenamiento indica que, con más *epochs*, quizás podría haber seguido aumentando. Sin embargo, se puede ver en el conjunto de validación que el modelo no estaba mejorando mucho, y sólo se hubiera conseguido

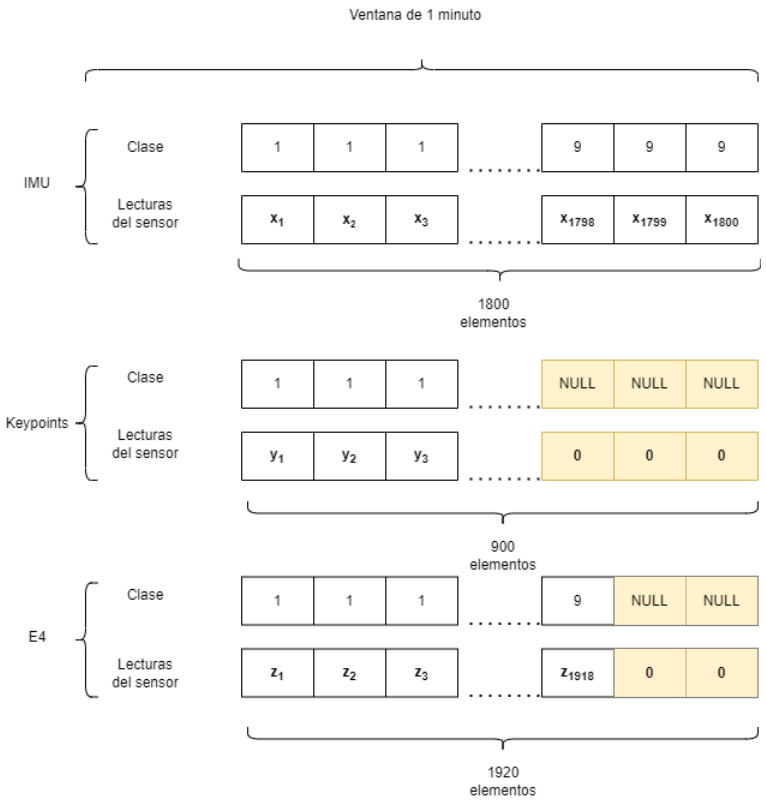

Figura 5.7: Preparación de los datos para la fusión por características

sobreajuste en caso de proseguir con el entrenamiento. Se observan en el conjunto de validación oscilaciones muy bruscas a lo largo del entrenamiento, y que a pesar de que la media de la precisión mejora, la pérdida empeora. La tasa de aprendizaje no es muy elevada, con lo que es posible que la red haya colapsado, aprendiendo a clasificar mejor las clases que están más presentes. En el conjunto de test, la fusión a nivel de características mejora sustancialmente la fusión a nivel de sensor, casi doblando la puntuación f1. La figura 5.8 detalla el caso del modelo con 64 unidades:

name	precision	recall	f1
avg/macro	0.398700	0.396640	0.387010
avg/weighted	0.442642	0.442381	0.437743
Picking	0.392100	0.292882	0.335337
Relocate Item Label	0.466287	0.379151	0.418228
Assemble Box	0.609342	0.675294	0.645062
Insert Items	0.240887	0.215603	0.227545
Close Box	0.349088	0.473034	0.401718
Attach Box Label	0.256651	0.455556	0.328328
Scan Label	0.757101	0.748306	0.752678
Attach Shipping Label	0.256724	0.180723	0.212121
Put on Back Table	0.227273	0.181159	0.201613
Fill out Order	0.341642	0.362928	0.351964

Figura 5.8: Resultados del modelo con fusión a nivel de características con 64 unidades en la capa de codificación en el conjunto de test

Para la opción con 80 unidades, las reflexiones son casi idénticas, excepto en que se aprecia una mejora, tanto en las curvas de entrenamiento como en la evaluación en el conjunto de test. En las figuras A.5 y A.6 aparece el desarrollo del entrenamiento, y en la figura 5.9 la puntuación final:

name	precision	recall	f1
avg/macro	0.433916	0.408312	0.415273
avg/weighted	0.472194	0.461082	0.462819
Picking	0.385526	0.341890	0.362400
Relocate Item Label	0.422376	0.518919	0.465696
Assemble Box	0.700275	0.599216	0.645816
Insert Items	0.310160	0.329078	0.319339
Close Box	0.319742	0.334831	0.327113
Attach Box Label	0.491228	0.311111	0.380952
Scan Label	0.790603	0.749274	0.769384
Attach Shipping Label	0.322734	0.349398	0.335537
Put on Back Table	0.274648	0.141304	0.186603
Fill out Order	0.321867	0.408100	0.359890

Figura 5.9: Resultados del modelo con fusión a nivel de características con 80 unidades en la capa de codificación en el conjunto de test

5.4. Fusión a nivel de decisión

Por la sencillez de su implementación, la fusión a nivel de decisión es una técnica frecuente en la literatura. Normalmente se asigna a cada clasificador un peso, posteriormente los clasificadores emiten sus veredictos acerca de la clase del modelo, y una estrategia de votación determina cual es la predicción final. El problema es que, en esta situación. la salida de los modelos no es una única clase, sino un vector con tantos elementos como haya en la secuencia, por lo tanto algunos sistemas de votación no son aplicables directamente, habría que (por ejemplo) recoger las votaciones de los modelos elemento a elemento. Otro punto importante es que la clase de los elementos que rodean a cada elemento en la secuencia puede dar información acerca de su propia clase, ya que las acciones se ejecutan habitualmente en un orden determinado. Para tomar en consideración ambos aspectos , he optado por una estrategia a nivel de decisión más conocida como *stacking*, técnica para combinar modelos de aprendizaje automático que funciona en dos fases: primero se entrenan modelos individuales por separado. En un segundo paso, y con los modelos base ya entrenados, se entrena un meta-modelo a partir de estos, cuya entrada serán las predicciones de cada uno de los modelos base, y su salida será la clasificación final.

El esquema del entrenamiento es similar al de la fusión por características, pero exige cuatro rondas de entrenamiento: una por cada modelo base y una última para el meta-modelo. Los modelos individuales se guardan según completen su entrenamiento. Para entrenar el meta-modelo, se cargan estos modelos individuales ya entrenados, se congelan sus pesos, y se añade al final de la arquitectura una red convolucional que se ocupa de la clasificación final.

5.4.1. Preprocesamiento de datos

El preprocesamiento de los datos es compartido con el de la fusión a nivel de características. El motivo es que, aunque el entrenamiento de cada modalidad individualmente solo necesite los datos del sensor que le corresponda, cuando se entrene el meta-modelo sí se requiere que reciba los datos de todos los sensores simultáneamente.

5.4.2. Evaluación

Ya que esta técnica de fusión implica entrenar primero los modelos base, se procederá a evaluar individualmente cada uno de ellos, y se terminará con el modelo general.

Evaluación del modelo entrenado sólo con datos del sensor IMU

Los resultados del entrenamiento del modelo utilizando únicamente datos del sensor IMU aparecen en las figuras A.7 y A.8. Siguen existiendo algunas oscilaciones en la precisión del modelo sobre el conjunto de validación a medida que se entrena, pero muchísimo más pequeñas. Esto sugiere que el problema podría estar en otra de las modalidades. Los resultados sobre el conjunto de test pueden verse en la figura 5.10 son también coherentes con la idea de que añadir otras de las modalidades perjudica la clasificación: se obtiene, sólo con los datos de un sensor, mejor puntuación que con cualquiera de las estrategias anteriores. Esto no quiere decir que fusionar por sensor o características no sea útil en general, sino que si hay problemas en los datos, o el modelo no es lo suficientemente complejo como para trabajar con una de las modalidades, los resultados serán peores que omitiendo la fusión.

name	precision	recall	f1
avg/macro	0.473836	0.447638	0.450371
avg/weighted	0.522807	0.489639	0.494049
Picking	0.418544	0.563594	0.480358
Relocate Item Label	0.522215	0.589961	0.554025
Assemble Box	0.779706	0.458039	0.577075
Insert Items	0.256711	0.434043	0.322615
Close Box	0.442771	0.330337	0.378378
Attach Box Label	0.588679	0.433333	0.499200
Scan Label	0.775070	0.800581	0.787619
Attach Shipping Label	0.383772	0.301205	0.337512
Put on Back Table	0.187702	0.210145	0.198291
Fill out Order	0.383193	0.355140	0.368634

Figura 5.10: Resultados del modelo entrenado sólo con datos del sensor IMU

Evaluación del modelo entrenado sólo con datos del sensor Kinect (keypoints)

En las figuras A.9 y A.10 se comprueba claramente qué modalidad era la que estaba dando problemas: el modelo entrenado sólo con datos provenientes de los puntos clave de Kinect no logra converger, existiendo una pequeña tendencia positiva a medida que se entrena, pero con

un competente demasiado alto del azar. Como se verá en la sección posterior, el entrenamiento con esta modalidad sí converge en el modelo desarrollado para competir, y de hecho, aunque también oscila según se entrena más que las otras dos modalidades, logra buenos resultados. Esto parece indicar que el modelo elegido no es capaz de aprender correctamente los datos de esta modalidad.

Evaluación del modelo entrenado sólo con datos del sensor E4

Por último se evalúa el sensor E4 en solitario. Lo que sucede en este caso, como se puede ver en las figuras A.11 y A.12 es que se da un sobrejauste a los datos de entrenamiento: la precisión en el conjunto de entrenamiento crece mientras que la precisión en el conjunto de validación disminuye, y la pérdida aumenta bruscamente. La figura 5.11 contiene los resultados de evaluar el modelo en el conjunto de test. Son peores que la fusión por características o que el sensor IMU en solitario, pero mejores que la fusión a nivel de sensor, lo que indica que la estrategia simple de interpolación que se siguió era errónea, bien por datos que no estuvieran alineados, o porque la variación entre mediciones es mucho mayor que la que predice la interpolación.

name	precision	recall	f1
avg/macro	0.321442	0.298964	0.299305
avg/weighted	0.373260	0.350644	0.352570
Picking	0.265227	0.401400	0.319406
Relocate Item Label	0.386935	0.416216	0.401042
Assemble Box	0.596899	0.543529	0.568966
Insert Items	0.205882	0.198582	0.202166
Close Box	0.295775	0.259551	0.276481
Attach Box Label	0.347656	0.247222	0.288961
Scan Label	0.648690	0.407551	0.500595
Attach Shipping Label	0.212871	0.148021	0.174619
Put on Back Table	0.071429	0.010870	0.018868
Fill out Order	0.183054	0.356698	0.241944

Figura 5.11: Resultados del modelo entrenado sólo con datos del sensor E4

Evaluación del meta-modelo

De acuerdo con [40], una buena característica de una combinación de clasificadores es que los clasificadores base tengan sus errores en clases distintas. De esa manera, los fallos de uno se ven compensados por los aciertos de otro, y se logra una mejor precisión final. En este caso, los datos del sensor IMU dominan a los de E4: no hay ninguna clase tal que sea clasificada con mejor precisión por IMU que por E4. La clase *Put On Back Table* es la más complicada para los dos clasificadores, con 0.198 de puntuación f1 para el sensor IMU, y 0.018 para E4. Si ordenáramos el resto de clases de mejor a peor clasificadas en ambos modelos, el resultado no sería exactamente igual, lo que puede ayudar en el entrenamiento del clasificador final.

En las figuras A.13 y A.14 se puede apreciar una de las características fundamentales de las combinaciones de modelos: reducen la varianza de los estimadores individuales. Las oscilaciones tan bruscas que se daban en el entrenamiento de los modelos base, debido a que el aprendizaje en una *epoch* pueda perjudicar a la siguiente, en el meta-modelo se moderan, siendo el aprendizaje

mucho más estable. Los resultados sobre el conjunto de test también son mejores, como se ve en la figura 5.12. El clasificador que combina los modelos base es el que mejor funciona hasta el momento, superando a cualquier otra combinación, y a los modelos individuales. Además, los resultados de incluir o no el modelo entrenado con los puntos clave no varían casi el resultado, lo que indica que el modelo aprende a obviar los clasificadores que no aportan información.

name	precision	recall	f1
avg/macro	0.492144	0.475851	0.480371
avg/weighted	0.530798	0.526788	0.525975
Picking	0.421156	0.501750	0.457934
Relocate Item Label	0.554582	0.635521	0.592299
Assemble Box	0.689365	0.706667	0.697909
Insert Items	0.318731	0.299291	0.308705
Close Box	0.485607	0.435955	0.459443
Attach Box Label	0.622568	0.444444	0.518639
Scan Label	0.786096	0.711520	0.746951
Attach Shipping Label	0.369208	0.425129	0.395200
Put on Back Table	0.266881	0.300725	0.282794
Fill out Order	0.407249	0.297508	0.343834

Figura 5.12: Resultados del meta-modelo que aprende a clasificar a partir de modelos individuales en el conjunto de test

5.5. Conclusiones

Las distintas técnicas aparecen comparadas en la figura 5.13. El mejor resultado se ha obtenido en la fusión a nivel de decisión, pero hay que destacar que este tipo de fusión tiene la desventaja de necesitar un entrenamiento mucho más exhaustivo, y tardar aproximadamente (para este caso particular) cuatro veces más que el resto, ya que se necesita entrenar los modelos base antes del clasificador final. Otro de los inconvenientes que se evitan con la fusión a nivel de decisión es el caso de que una de las modalidades dé problemas con el modelo elegido, porque al separar las modalidades entre sí los problemas que cause una de ellas no se propagan a las demás. En resumen, la fusión a nivel de decisión tiene las ventajas de ser la más sencilla de implementar, de reducir de manera natural la dispersión entre los clasificadores y contener los problemas que puedan dar modalidades individualmente, pero requiere mayor esfuerzo en la implementación al necesitarse cuatro modelos, y el modelo resultante será por tanto cuatro veces mayor.

La fusión a nivel de características es la más popular en la literatura. No ha desempeñado en este caso mucho peor que a nivel de decisión, pero es más difícil de implementar. Requiere una arquitectura más compleja que permita unir las características de cada una de las modalidades en algún paso intermedio, y elegir dónde y cómo realizar esas unión no es trivial, y una mala elección puede dar lugar a modelos subóptimos.

Por último, la fusión a nivel de sensor es la que peor resultados ha dado. Ha resultado muy sensible a los datos, y además cuando en este caso no se pueden fusionar directamente las lecturas, al pertenecer a instantes de tiempo distintos por la diferente tasa de muestreo, implica reducir la información disponible, o inventar nueva.

Respecto al modelo elegido, no ha cumplido las expectativas. Puede que sea debido a que no es lo suficientemente potente, o a errores en la implementación particular desarrollada para el proyecto, pero lo cierto es que la puntuación obtenida ha sido baja. No era el objetivo de esta sección obtener un número alto, sino simplemente comparar técnicas de fusión, y para ello sí ha resultado valioso. En la siguiente sección se tendrán en cuenta las conclusiones para intentar crear un modelo que obtenga una puntuación mejor.

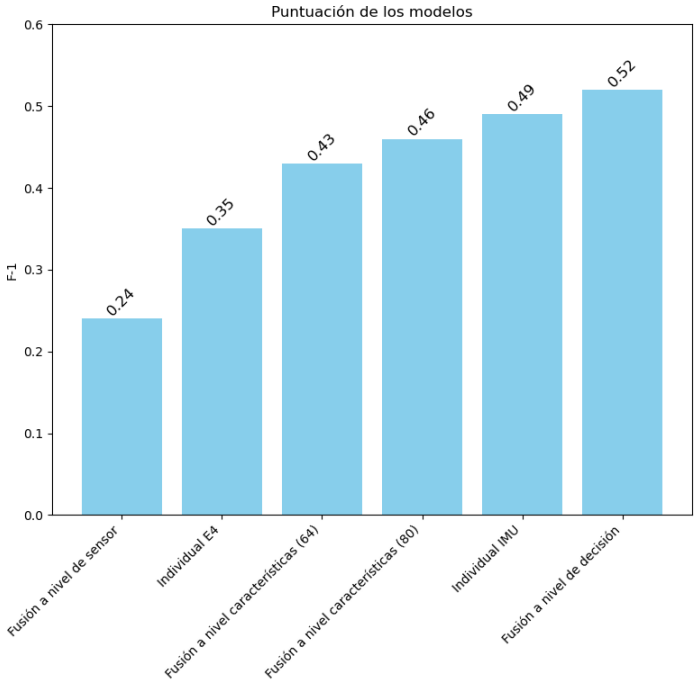

Figura 5.13: Puntuación de las distintas técnicas y modelos

Capítulo 6

Desarrollo del modelo para la competición

El objetivo a partir de ahora es desarrollar un modelo que sea capaz de obtener una puntuación alta, como si estuviera participando realmente en la competición. De la sección anterior se han podido sacar las conclusiones de que la fusión a nivel de decisión funciona mejor, y que el modelo elegido probablemente no sea el adecuado, ya sea por ser teóricamente insuficiente o por no haber sido implementado correctamente. Por tanto, se va a entrenar modelos individuales usando cada una de las modalidades con un arquitectura distinta, y se van a fusionar los resultados a nivel de decisión.

6.1. Arquitectura elegida

El modelo elegido viene detallado en Challa, Kumar y Semwall [10]. La diferencia de la arquitectura propuesta con otras de la literatura consiste en ramificar el procesamiento de los datos. A la entrada al modelo, cada entrada de datos (en este caso una secuencia) se separa en tres ramas distintas. Cada una de las ramas consiste en dos redes de convolución, junto a capas de agregación. Las ramas se diferencian entre sí por el tamaño del kernel de sus convoluciones: en la implementación tienen tamaños de 1, 5 y 10. De este modo, cada una de las ramas estaría aprendiendo características locales a un nivel distinto de abstracción: elemento a elemento, en grupos de 5 elementos, o en grupos de 10. La idea es que los patrones que se puedan encontrar en los datos pueden darse en tamaños distintos.

Una vez que cada una de las tres ramas ha extraído las características propias, los tensores resultantes se concatenan, y pasan a dos capas LSTM bidireccionales. EL motivo de usar LSTM en este paso es poder detectar las dependencias temporales de las características entre sí, y al usar LSTM bidireccionales el modelo adquiere la capacidad de aprender estas relaciones también cuando elementos posteriores en el tiempo informan acerca de la clase de elementos anteriores.

En la figura 6.1 se muestra el esquema de la arquitectura. La justificación de porqué este modelo ha sido elegido es que, además de obtener buenas puntuaciones en otros conjuntos de datos, incluye LSTM (cuya ausencia es quizás uno de los problemas en el modelo anterior), y su arquitectura es sencilla de implementar. La implementación del modelo del proyecto difiere en algunos puntos de la arquitectura original:

1. Se han añadido capas de normalización por lotes a las capas de convolución, para incluir el pre-procesamiento de los datos en el propio modelo.

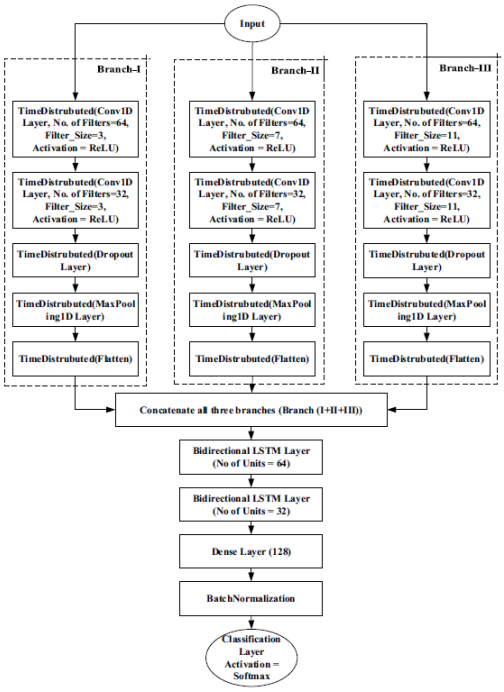

Figura 6.1: Esquema de la arquitectura en la que se basó el modelo competitivo. Fuente original [10]

2. Se ha añadido *dropout* a continuación de las capas LSTM.

3. Como ya se explicó en el capítulo previo, se ha utilizado una capa convolucional en lugar de una completamente conectada para la salida del modelo.

Las siguientes secciones consistirán en la explicación de los modelos individuales, entrenados cada uno de ellos con una única modalidad. El capítulo se cierra con la explicación de la fusión de modelos, en una organización similar a la del capítulo previo.

6.2. Preprocesamiento de los datos

La preparación de los datos previa al entrenamiento es la misma que se utilizó con la arquitectura CSNet. Lo único distinto es el modelo, y no es necesario introducir ningún cambio en la fase de preprocesamiento, lo que permite reciclar todo el código desarrollado anteriormente. Reiterando en lo explicado en el capítulo anterior, el entrenamiento se desarrolla en dos etapas: la primera consiste en entrenar individualmente y por separado tres modelos distintos, en el que cada uno se corresponde con una modalidad. La segunda, en el entrenamiento del metamodelo que usa las predicciones de estos clasificadores base. Las figuras 6.2 y 6.3 contienen los diagramas de estas etapas respectivamente.

6.3. Modelos individuales

6.3.1. Modelo con datos de IMU

La diferencia fundamental con respecto a este sensor ha sido incluir aquí también los datos del giroscopio. En el modelo anterior, al no interesar obtener la mejor puntuación posible, se

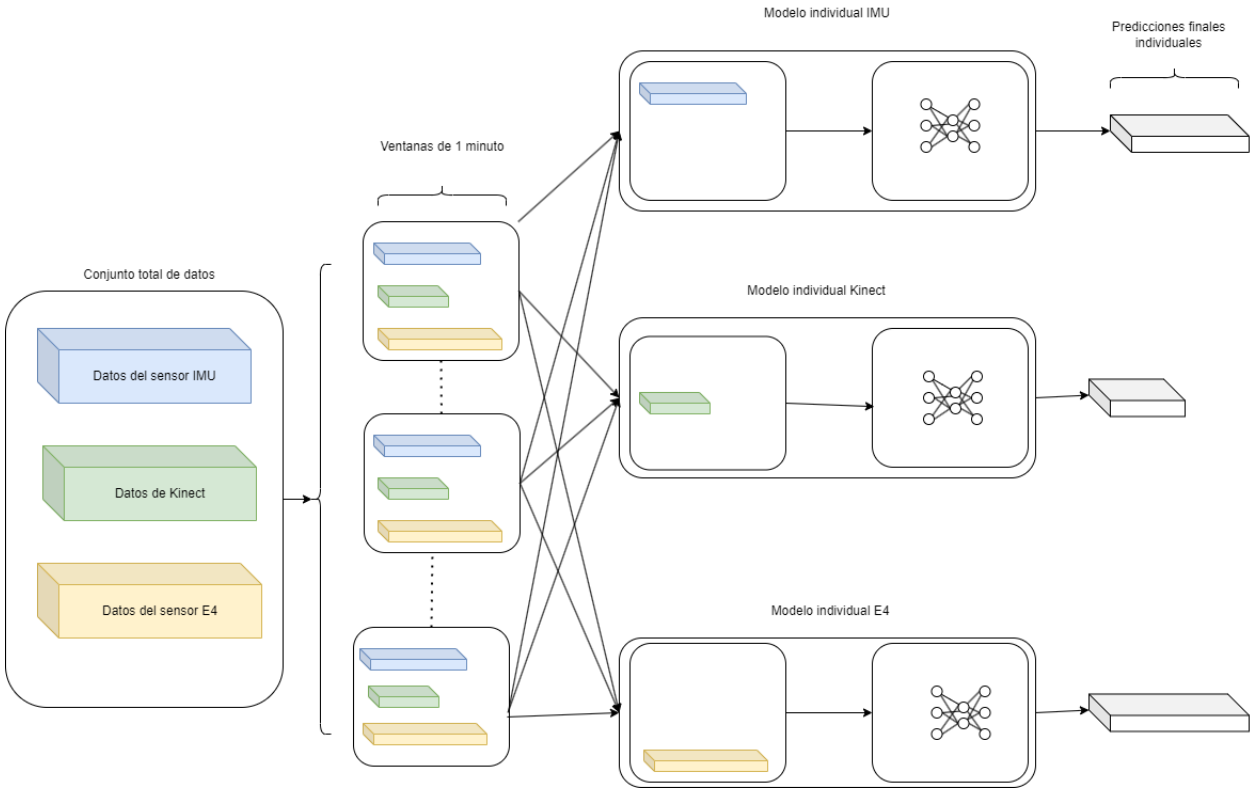

Figura 6.2: Etapa 1: Entrenamiento de los modelos base

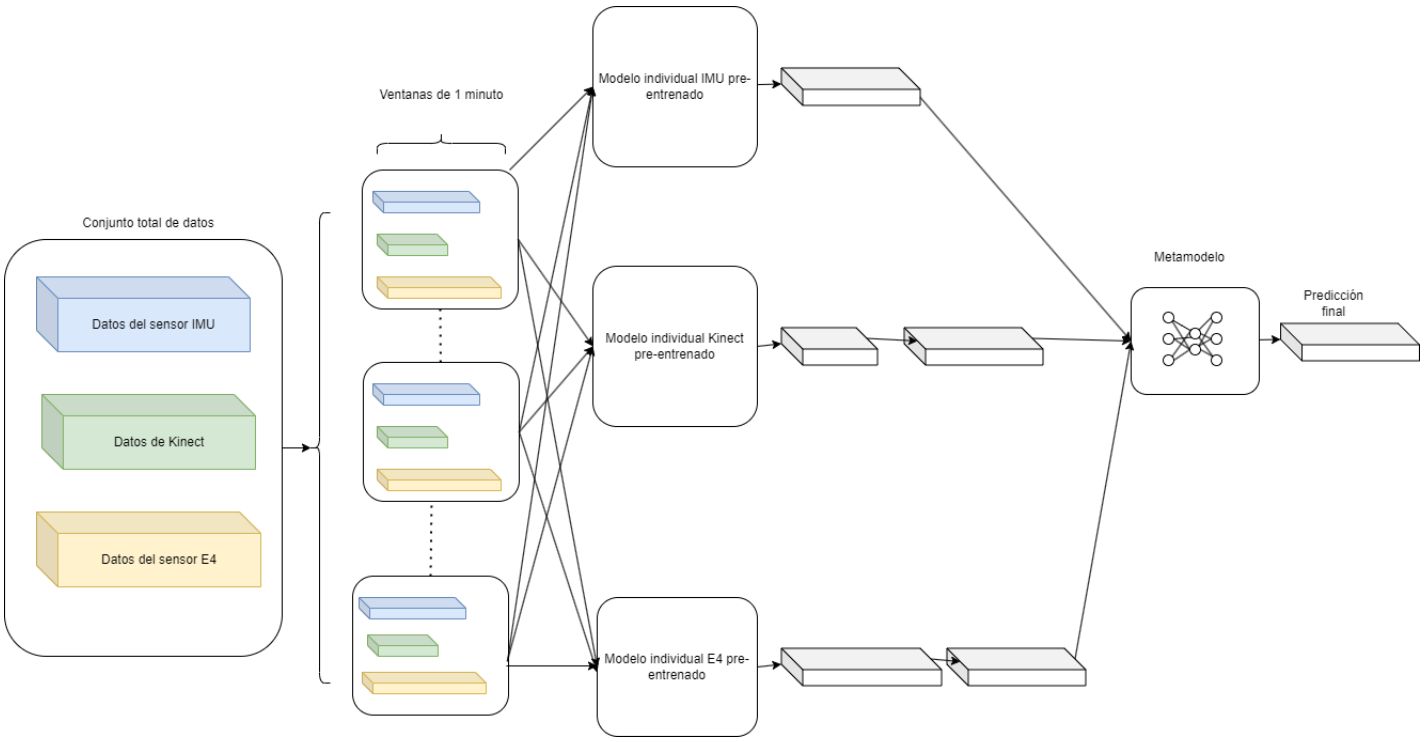

Figura 6.3: Etapa 2: Entrenamiento del meta-modelo a partir de los modelos base.

añadió la cantidad de información configurada por defecto (según la propia organización) para desarrollarlo. Ahora, cuanto mayor información esté disponible, mejor podrá (en principio)

rendir el modelo. También existe la posibilidad de añadir la información dada por el cuaternión, pero se indica en la documentación del conjunto de datos que existen errores en algunas de las mediciones, así que decidí no incluirlo. En las figuras A.15 y A.16 se puede comprobar la gran mejora que tanto la información del giroscopio como la nueva arquitectura supusieron para la clasificación. El entrenamiento fue además muy estable, sin apenas oscilaciones en los valores de precisión y pérdida. El resultado en el conjunto de test aparece en la figura 6.4, siendo el mejor hasta el momento.

name	precision	recall	f1
avg/macro	0.916318	0.902584	0.908520
avg/weighted	0.917258	0.915719	0.915675
Picking	0.927607	0.882147	0.904306
Relocate Item Label	0.872207	0.964479	0.916025
Assemble Box	0.944308	0.904314	0.923878
Insert Items	0.895105	0.907801	0.901408
Close Box	0.953271	0.916854	0.934708
Attach Box Label	0.836957	0.855556	0.846154
Scan Label	0.935814	0.973863	0.954459
Attach Shipping Label	0.945455	0.895009	0.919540
Put on Back Table	0.970833	0.844203	0.903101
Fill out Order	0.881620	0.881620	0.881620

Figura 6.4: Resultados en el conjunto de test del modelo para la competición, entrenado únicamente con los datos IMU

6.3.2. Modelo con datos de Kinect

El entrenamiento del modelo con los datos de Kinect continua sufriendo muchas oscilaciones, como muestran las figuras A.17 y A.18, pero esta vez sí logra que la precisión crezca y la pérdida disminuya en el conjunto de validación. De todos modos, la inestabilidad sigue siendo demasiado alta incluso con la nueva arquitectura, lo cual parece indicar ya definitivamente que es necesario un pre-procesamiento más profundo de los datos para que puedan ser realmente útiles, si eso fuera posible.

6.3.3. Modelo con datos de E4

El entrenamiento con los datos del sensor E4 con esta nueva arquitectura es mucho más estable, y además no da lugar a sobreajuste, como ocurrió con el modelo CSNet. En las figuras A.19 y A.20 aparece su desarrollo. También en el conjunto de test la puntuación obtenida por este modelo es superior a la del anterior, como puede verse en la figura 6.5, pero sin embargo se queda bastante lejos de la lograda por el modelo entrenado con los datos IMU.

6.4. Meta-modelo

Una vez se han entrenado los modelos individuales, se puede realizar el entrenamiento del meta-modelo. El primer problema que enfrenta este entrenamiento es que las predicciones de

name	precision	recall	f1
avg/macro	0.712306	0.657982	0.664663
avg/weighted	0.729835	0.677913	0.682857
Picking	0.693878	0.674446	0.684024
Relocate Item Label	0.847107	0.633205	0.724702
Assemble Box	0.806695	0.756078	0.780567
Insert Items	0.837037	0.641135	0.726104
Close Box	0.844197	0.596629	0.699144
Attach Box Label	0.758621	0.488889	0.594595
Scan Label	0.458080	0.946757	0.617424
Attach Shipping Label	0.597096	0.566265	0.581272
Put on Back Table	0.548969	0.771739	0.641566
Fill out Order	0.731377	0.504673	0.597235

Figura 6.5: Resultados en el conjunto de test del modelo para la competición, entrenado únicamente con los datos del sensor E4

cada uno de los modelos tienen distintas dimensiones, porque la tasa de muestreo de cada sensor es distinta, y por tanto en cada ventana de tiempo cada modelo emite un número distinto de predicciones. La solución seguida es escoger una de las modalidades para mantener inalterada, y hacer *up/downsampling* de cada uno de los vectores con las predicciones del resto de modalidades. El *up/downsampling* consiste en interpolar linealmente los valores de un vector para que tengan el tamaño deseado, pasando, por ejemplo, de los 1800 elementos de la modalidad IMU a 1920, que son los que tiene la modalidad E4, consiguiendo unificar las dimensiones de los vectores con las predicciones que provienen de los distintos sensores.

Las figuras 6.6 y 6.7 contienen los resultados de la evaluación de los dos tipos de modelo generados: en el primero se mantuvieron las predicciones del sensor E4 (y por tanto se modifcó el tamaño de IMU y Kinect), y en el segundo se mantuvo el tamaño de IMU.

name	precision	recall	f1
avg/macro	0.795652	0.765669	0.776294
avg/weighted	0.796001	0.791130	0.789513
Picking	0.733405	0.799300	0.764936
Relocate Item Label	0.813733	0.796139	0.804840
Assemble Box	0.809524	0.893333	0.849366
Insert Items	0.852649	0.730496	0.786860
Close Box	0.868644	0.691011	0.769712
Attach Box Label	0.717949	0.544444	0.619273
Scan Label	0.715283	0.892546	0.794143
Attach Shipping Label	0.789474	0.800344	0.794872
Put on Back Table	0.842105	0.753623	0.795411
Fill out Order	0.813758	0.755452	0.783522

Figura 6.6: Resultados del meta-modelo, manteniendo el tamaño de las predicciones de la modalidad E4, y modificando las demás.

name	precision	recall	f1
avg/macro	0.911321	0.909679	0.910197
avg/weighted	0.918700	0.918120	0.918152
Picking	0.904815	0.942824	0.923429
Relocate Item Label	0.932602	0.918919	0.925710
Assemble Box	0.917582	0.916863	0.917222
Insert Items	0.878342	0.931915	0.904336
Close Box	0.961268	0.920225	0.940299
Attach Box Label	0.828418	0.858333	0.843111
Scan Label	0.940228	0.959342	0.949689
Attach Shipping Label	0.927434	0.901893	0.914485
Put on Back Table	0.916981	0.880435	0.898336
Fill out Order	0.905537	0.866044	0.885350

Figura 6.7: Resultados del meta-modelo, manteniendo el tamaño de las predicciones de la modalidad IMU, y modificando las demás.

Como se puede ver, en ambos casos la fusión a nivel de decisión mejora la puntuación con respecto al clasificador cuyas predicciones se dejaron intactas, aunque en el caso de IMU es casi inapreciable. Hay un detalle importante: a pesar de que la fusión a nivel de decisión cuando se deja el vector de E4 intacto mejora la puntuación obtenida con respecto al modelo base entrenado con E4, no es mejor que el modelo base entrenado con los datos IMU. Esto sugiere que al redimensionar el vector de predicciones se pierde algo de información, aunque no completamente ya que en ese caso no habría mejora alguna. El meta-modelo en el que se mantuvieron las predicciones de los datos IMU intactas ha sido el mejor de todos los desarrollados, superando por centésimas el modelo entrenado sólo con los datos IMU. A pesar de que el incremento en precisión es muy pequeño, las figuras A.21 y A.22 indican que tiene muchísima menos dispersión debido a la agregación de resultados de los distintos modelos, lo que puede implicar mejor capacidad de generalización.

6.5. Evaluación de resultados

Los resultados del modelo para la competición son bastante positivos. En la tabla 6.1 se pueden ver las puntuaciones de todos los participantes. Esas puntuaciones no tienen porqué ser las mismas que las obtenidas en el conjunto de test público, pero dado que la competición ya ha concluido es la única forma que hay de comparar la solución propuesta con la del resto de participantes. Teniendo esto en cuenta, el modelo desarrollado habría quedado en cuarta posición, cerca de las posiciones dotadas con premio económico (las tres primeras), y entre las cinco primeras, que fueron premiadas con un viaje costeado a las conferencias en cuyo marco se planteó el desafío

Equipo	Puntuación F1
tomoon	0.9633
vbu211	0.9592
Ritsumei	0.9241
Enrique Pérez	0.9181
Malton	0.9171
Shubham Wagh	0.9112
UCLab	0.9057
liuqijd	0.8987
Potros	0.8822
Dialga	0.8752

Cuadro 6.1: Top 10 de solcuciones para el Open Pack Challenge, incluyendo el proyecto desarrollado entre ellas

Respecto a las técnicas utilizadas para lograr la puntuación obtenida, creo que lo más importante ha sido el modelo elegido. Ramificar los datos de entrada entre varias capas convolucionales que trabajen en paralelo para poder extraer datos a distintas resoluciones (dependiendo del tamaño del kernel de las capas de convolución en cada una de las ramas) es especialmente útil en problemas como este, ya que las actividades varían en granularidad, es decir, que algunas están determinadas por secuencias más cortas de eventos que otras. De entre las soluciones previas, la que originalmente quedó en quinta posición [1] presentó un modelo sin fusión, utilizando únicamente el sensor IMU y agregando tanto aumento de datos como un mecanismo de atención posterior a las capas de convolución, y a las unidades LSTM. La arquitectura es similar, pero el añadir la bifurcación en los datos ha otorgado al modelo presentado un incremento en puntuación f1 superior al otorgado por el aumento de datos y por la atención.

Otro factor a tener en cuenta es la fusión. No ha sido especialmente relevante en el modelo final, pero sí ha ayudado a conseguir unas pocas centésimas, sin las cuales la solución quedaría en quinto lugar. La solución ganadora[2] realiza la fusión a nivel de características: primero hay una serie de capas de *embedding* que aprenden las características de las secuencias de las modalidades, luego esas capas se concatenan, y se procesan en una capa completamente conectada que aprende una codificación de menor dimensión que la del tensor que forman las características de todas las modalidades concatenadas entre sí. Esta es una técnica mucho más elaborada que la propuesta, y por tanto más complicada de desarrollar, pero ha logrado el primer puesto. La solución en segundo lugar[3] realiza una fusión que es a la vez de puntuación y de características: el clasificador final recibe como entradas tanto las puntuaciones de los clasificadores base como las características extraídas en pasos anteriores. Una vez más, un desarrollo así es mucho más complejo de llevar a cabo, algo que hay que valorar teniendo en cuenta el posible margen de mejora con respecto a una solución más sencilla. El tipo de fusión que emplea la solución que quedó en tercera posición[4] es a nivel de puntuación, el más sencillo y también el más parecido al propuesto en el proyecto.

El problema de la fusión, a cualquier nivel, es el utilizar sensores con diferentes tasas de muestreo. En algún punto del proceso de clasificación hay que modificar el tamaño de alguno de los tensores, para que pueda hacerse la fusión en ese punto. En el caso de realizarla a nivel de

[1] https://open-pack.github.io/data/2023-03-13-openpack-challenge-poster/5th-shubham.pdf
[2] https://open-pack.github.io/data/2023-03-13-openpack-challenge-poster/1st-tomoon.pdf
[3] https://open-pack.github.io/data/2023-03-13-openpack-challenge-poster/2nd-vbu211.pdf
[4] https://open-pack.github.io/data/2023-03-13-openpack-challenge-poster/3rd-ritsumei.pdf

decisión, esta redimensión se hace con los vectores de salida de los modelos base, que contienen las predicciones, lo que provoca una distorsión en la salida y una pérdida de información aunque la interpolación sea lineal. En cualquier caso, ayuda a estabilizar la salida del modelo, pues aunque las distorsiones en las puntuaciones de cada clase al interpolar puedan dar lugar a clasificaciones erróneas, estas puntuaciones, una vez el modelo ha sido entrenado, están centradas en torno a la clase correcta, por tanto al sumar los resultados de varias modalidades entre sí normalmente la clase correcta alcanza un valor más alto. De entre las soluciones previas, la ganadora utiliza un complejo algoritmo para conseguir que todas las lecturas tengan la misma longitud, es decir, la redimensión se hace a nivel de sensor. La solución en tercera posición delega esa lógica a la etapa de fusión por puntuación de los clasificadores base, es decir en el mismo paso en el que se realiza en el presente proyecto.

También fue muy importante agregar la información del giroscopio a la del acelerómetro, pasando de 0.75 a 0.92 en la puntuación f1 sólo con ese cambio. La solución en cuarto puesto[5] presenta una arquitectura muy similar a la del proyecto, pero no incluye los datos del giroscopio al procesar los datos IMU. Es posible que tan sólo realizando el pequeño cambio de añadirlos hubiera conseguido una posición mejor, en vista de la mejora que supuso para el modelo desarrollado.

En resumen, las soluciones que quedaron en las tres primeras posiciones implementaron lógicas mucho más elaboradas tanto a nivel de pre-procesamiento de los datos, como en la arquitectura de clasificación y en la estrategia de fusión. Las soluciones originalmente cuarta y quinta no son muy diferentes de la propuesta en el proyecto, pero ambas carecen de algún aspecto (inclusión del giroscopio o procesamiento paralelo, respectivamente), que podría haber mejorado su puntuación.

[5] https://open-pack.github.io/data/2023-03-13-openpack-challenge-poster/4th-malton.pdf

Capítulo 7

Conclusiones y trabajos futuros

En este capítulo se presentan las conclusiones a las que se han llegado tras finalizar el trabajo, y las posibles mejoras que podrían hacerse en un futuro. Respecto a los objetivos marcados, los dos han sido cumplidos: se ha explorado de forma sistemática las posibles estrategias de fusión de modalidades, y se ha desarrollado un modelo que podría haber alcanzado una posición alta en el desafío (aún insistiendo en que los datos con los que se han evaluado las soluciones oficiales y en el presente proyecto no son los mismos).

En el primer objetivo, el de fusión de modalidades, el principal problema es usar sensores con tasas de muestreo distintas. Esto hace que para fusionar la información en algún punto haya que cambiar las dimensiones de la salida de alguno de ellos. Si este cambio se hace a nivel de sensor, en este conjunto de datos particular al menos, se introducen demasiadas distorsiones en los datos. A nivel de características es dónde más opciones hay, pero es una técnica más difícil de implementar. Por último, a nivel de decisión, cambiar el tamaño de los vectores de salida interpolando los datos puede introducir distorsiones muy grandes en la clasificación final, pues lo que se interpola aquí no son mediciones del sensor, o características intermedias, sino directamente las puntuaciones de la clase. Tiene la ventaja de ser sencillo de implementar por un lado, y por otro que, al ser necesario entrenar los modelos individuales, si alguno de ellos proporciona una precisión que es suficiente para el cometido que se busca, no es necesario continuar con la implementación. En este caso, el modelo entrenado únicamente con el sensor IMU logró ya una puntuación suficientemente alta, por tanto en el caso de estar en un proyecto con restricciones temporales podría haberse detenido el desarrollo ahí. Dicho en otras palabras, si se opta por la fusión a nivel de puntuación, es muy probable que no se pierda el tiempo en caso de que la fusión no funcione, porque ya se tienen los modelos individuales entrenados. Esto por otra parte, es la gran desventaja de esta técnica: la necesidad de entrenar los modelos base por separado puede consumir muchos recursos.

Del segundo objetivo creo que no se puede agregar nada más a lo expuesto cuando se evaluaban los resultados en el capítulo anterior, excepto una consideración acerca de la complejidad de las soluciones publicadas. Varias de ellas utilizan un preprocesamiento de los datos muy elaborado, y/o arquitecturas mucho más complejas que la propuesta, lo cual se traduce en unas décimas de mejora en la puntuación F1. Este es un detalle a tener en cuenta en proyectos de estas características: hasta que punto merece la pena invertir esfuerzo en desarrollar un modelo que logre puntuaciones cada vez más altas es algo que se tiene que valorar caso por caso, y quizás la respuesta sea negativa a veces.

7.1. Trabajos futuros

El tamaño del conjunto de datos ha sido una restricción muy importante a la hora de elaborar el presente trabajo. Debido al tiempo que se tarda en completar un entrenamiento (alrededor de las 3 horas para 100 *epochs*), ajustar los hiperparámetros para intentar obtener una mejor solución ha sido imposible. La estrategia seguida ha sido la de un algoritmo voraz: se probaron una serie de valores en una serie de hiperparámetros, y se elige el valor que mejor funcione, pero sin volver atrás y sin probar las distintas combinaciones entre ellos. Como ejemplo de los hiperparámetros que puede que hubieran dado lugar a una mejor solución, pueden modificarse (en muchos otros):

1. **Optimizador:** se utilizó Adam
2. **Tasa de aprendizaje:** La elegida fue 0.01, descartándose después de probar en un modelo 0.02 y 0.001.
3. **Tamaño de los kernels de las redes de convolución en la arquitectura utilizada para la competición:** se probaron tanto los valores indicados en el artículo original como 1/5/10, siendo mejor este último.
4. **Tamaño del kernel de la capa de salida:** Al ser la capa encargada de la clasificación también una capa convolucional, esta puede tener distintos tamaños de kernel, lo cuál significaría que las puntuaciones de clase dependen también de los elementos siguientes en la secuencia. Se probaron los valores 1 y 3, no existiendo importantes diferencias entre ambos.

Otro aspecto que puede ser mejorado en el trabajo es hacer una preparación de los datos más exhaustiva. Algunas medidas de los sensores no se han incluido por existir errores en los datos, sin embargo, en lugar de simplemente descartarlas, puede hacerse un proceso de sustituir los datos que sean erróneos o corruptos por datos imputados, pudiendo así añadir el resto que sí son correctos.

Pero el aspecto más importante que se puede hacer para mejorar un proyecto de estas características es realizar un trabajo más detalladoa redimensionando los datos para que los modelos tengan la misma salida, sea cual sea la tasa de muestreo del sensor correspondiente. En el código desarrollado simplemente se efectúo una interpolación, pero se ha comprobado que no es suficiente, y que introduce distorsiones en los datos. Otras alternativas serían imputar los datos teniendo en cuenta información estadística de la secuencia, o añadiendo capas de convolución inversa que pudieran aprender la forma adecuada de aumentar el tamaño de un tensor.

Bibliografía

[1] A. A. Aguileta, R. F. Brena, L. A. Trejo, E. Molino-Minero-Re, O. Mayora, and L. A. Trejo. Virtual sensors for optimal integration of human activity data. *Computational Intelligence-Based Sensors*, 2019.

[2] M. H. Arshad, M. Bilal, and A. Gani. Human activity recognition: Review, taxonomy and open challenges. *Sensors*, 22(17), 2022.

[3] Binjal B. Suthar and B. Gadhia. Human activity recognition using deep learning: A survey. In Ketan Kotecha, Vincenzo Piuri, Hetalkumar N. Shah, and Rajan Patel, editors, *Data Science and Intelligent Applications*, pages 217–223, Singapore, 2021. Springer Singapore.

[4] A. Bagnall, J. Lines, A. Bostrom, J. Large, and E. Keogh. The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery*, 2016.

[5] O. Banos, R. Garcia, and A. Saez. MHEALTH Dataset. UCI Machine Learning Repository, 2014. DOI: https://doi.org/10.24432/C5TW22.

[6] E. A. Bernal, X. Yang, Q. Li, J. Kumar, S. Madhvanath, P. Ramesh, and R. Bala. Deep temporal multimodal fusion for medical procedure monitoring using wearable sensors. *IEEE Transactions on Multimedia*, 20:107–118, 2018.

[7] E. Blasch, A. Vakil, J. Li, and R. Ewing. Multimodal data fusion using canonical variates analysis confusion matrix fusion. In *2021 IEEE Conference on Aerospace*, 2021.

[8] R. F. Brena, A. A. Aguileta, L. A. Trejo, E. Molino-Minero-Re, and O. Mayora. Choosing the best sensor fusion method: A machine-learning approach. *Mobile Sensors for Healthcare*, 2020.

[9] M. Burchi and R. Timofte. Audio-visual efficient conformer for robust speech recognition, 2023.

[10] S. Challa, A. Kumar, and V. Semwal. A multibranch cnn-bilstm model for human activity recognition using wearable sensor data. *The Visual Computer*, 38, 08 2021.

[11] M. Dhawan and T. Aggrawi. State-of-the-art vs prominent models: An empirical analysis of various neural networks on stock market prediction. *Independent Scholar*, 2022.

[12] J. Dou, J. Xue J., and Fang. Seg-voxelnet for 3d vehicle detection from rgb and lidar data. pages 4362–4368, 05 2019.

[13] E. Essa and I. R. Abdelmaksoud. Temporal-channel convolution with self-attention network for human activity recognition using wearable sensors. *Knowledge-Based Systems*, 278:110867, 2023.

[14] Z. Feng, F. Jiang, and R. Shen. Virtual glasses try-on based on large pose estimation. *Procedia Computer Science*, 131:226–233, 01 2018.

[15] M. Greco, M. Spagnoletta, A. Appice, and D. Malerba. Applying machine learning to predict closing prices in stock market: A case study. In *Mining Data for Financial Applications: 5th ECML PKDD Workshop*, 2021.

[16] A. H. Gunatilaka and B. A. Baertlein. Feature-level and decision-level fusion of noncoincidently sampled sensors for land mine detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2001.

[17] N. Gupta, S. K. Gupta, R. K. Pathak, V. Jain, P. Rashidi, and J. S. Suri. Human activity recognition in artifcial intelligence framework: a narrative review. *Artifcial Intelligence Review*, 2022.

[18] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2015.

[19] I. K. Ihianle, A. O. Nwajana, S. H. Ebenuwa, R. I. Otuka, K. Owa, and M. O. Orisatoki. A deep learning approach for human activities recognition from multimodal sensing devices. *IEEE Access*, 8:179028–179038, 2020.

[20] R. Karthikeswaren, K. Kayathwal, G. Dhama, and A. Arora. A survey on classical and deep learning based intermittent time series forecasting methods. In *International Joint Conference on Neural Networks (IJCNN)*, 2021.

[21] R. Kavi, V. Kulathumani, F. Rohit, and V. Kecojevic. Multi-view fusion for activity recognition using deep neural networks. 2016.

[22] A. Khan, N. Hammerla, S. Mellor, and T. Ploetz. Optimising sampling rates for accelerometer-based human activity recognition. *Pattern Recognition Letters*, 73, 01 2016.

[23] M.E. Ligins, D.L. Hall, and J. Llinas. *Handbook of Multisensor Data Fusion: Theory and Practice*. CRC Press: Boca Raton, FL, USA, 2008.

[24] Z. C. Lipton, D. C. Kale, C. P. Elkan, and R. C. Wetzel. Learning to diagnose with lstm recurrent neural networks. *CoRR*, abs/1511.03677, 2015.

[25] I. D. Luptáková, M. Kubovčík, and J. Pospíchal. Wearable sensor-based human activity recognition with transformer model. *Sensors*, 22(5), 2022.

[26] N. Ma, C. Xiao, M. Wang, and G. Xu. A review of point cloud and image cross-modal fusion for self-driving. In *2022 18th International Conference on Computational Intelligence and Security (CIS)*, pages 456–460, 2022.

[27] mmpose. `https://github.com/open-mmlab/mmpose`.

[28] M. Mohandes, M. Deriche, and S. O. Aliyu. Classifiers combination techniques: a comprehensive review. *IEEE Access*, 2018.

[29] OmegaConf. `https://omegaconf.readthedocs.io/en/2.3_branch/`.

[30] F. J. Ordoñez and D. Fischer. Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition. *Sensors (Basel, Switzerland)*, 2016.

[31] S. Raschkaand, Y. Liu, V. Mirjalili, and D. Dzhulgakov. *Machine Learning with PyTorch and Scikit-Learn: Develop machine learning and deep learning models with Python*. Packt Publishing, 2022.

[32] A. Reiss. PAMAP2 Physical Activity Monitoring. UCI Machine Learning Repository, 2012. DOI: https://doi.org/10.24432/C5NW2H.

[33] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. *ArXiv*, abs/1505.04597, 2015.

[34] G. Shao, Y. Chen, and Y. Wei. Deep fusion for radar jamming signal classification based on cnn. *IEEE Access*, 8:117236–117244, 2020.

[35] E. Shelhamer, J. Long, and T. Darrell. Fully convolutional networks for semantic segmentation. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440, 2014.

[36] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[37] S. P. Singh, M. K. Sharma, A. Lay-Ekuakille, D. Gangwar, and S. Gupta. Deep ConvLSTM with self-attention for human activity decoding using wearable sensors. *IEEE Sensors Journal*, 21(6):8575–8582, mar 2021.

[38] P. Soares, A. Silva, and L. Pereira. An assault detection system based on human pose tracking for video surveillance. pages 192–194, 10 2019.

[39] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, 2014.

[40] A. Tsymbal, M. Pechenizkiy, and P. Cunningham. Diversity in search strategies for ensemble feature selection. *Information Fusion*, 6(1):83–98, 2005. Diversity in Multiple Classifier Systems.

[41] A. Vaswani, N. M. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L.z Kaiser, and I. Polosukhin. Attention is all you need. In *NIPS*, 2017.

[42] K. Vyas, R. Ma, B. Rezaei, S. Liu, M. Neubauer, T. Ploetz, R. Oberleitner, and S. Ostadabbas. Recognition of atypical behavior in autism diagnosis from video using pose estimation over time. In *IEEE 29th International Workshop on Machine Learning for Signal Processing*, pages 1–6, 10 2019.

[43] J. Wang, Z. Wei, T. Zhang, and W. Zeng. Deeply-fused nets. *ArXiv*, abs/1605.07716, 2016.

[44] G. M. Weiss. Wisdm smartphone and smartwatch activity and biometrics dataset. *UCI Machine Learning Repository, WISDM Smartphone and Smartwatch Activity and Biometrics Dataset Data Set*, 2019.

[45] R. Wirth and J. Hipp. Crisp-dm: Towards a standard process model for data mining. *Proceedings of the 4th International Conference on the Practical Applications of Knowledge Discovery and Data Mining*, 01 2000.

[46] Y. Wu, P. Peng, Z. Zhang, Y. Zhao, and B. Qin. An efficient end-to-end transformer with progressive tri-modal attention for multi-modal emotion recognition, 2022.

[47] B. Yu, Y. Liu, and K. Chan. Survey of sensor modalities for human activity recognition. In *Proceedings of the 12th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2020) - Volume 1: KDIR, pages 282-294*, 2020.

[48] Z. Zhang and B. Wang. Graph spring network and informative anchor selection for session-based recommendation. *Neural networks : the official journal of the International Neural Network Society*, 159:43–56, 2022.

Apéndice A

Gráficas con los resultados de la evaluación

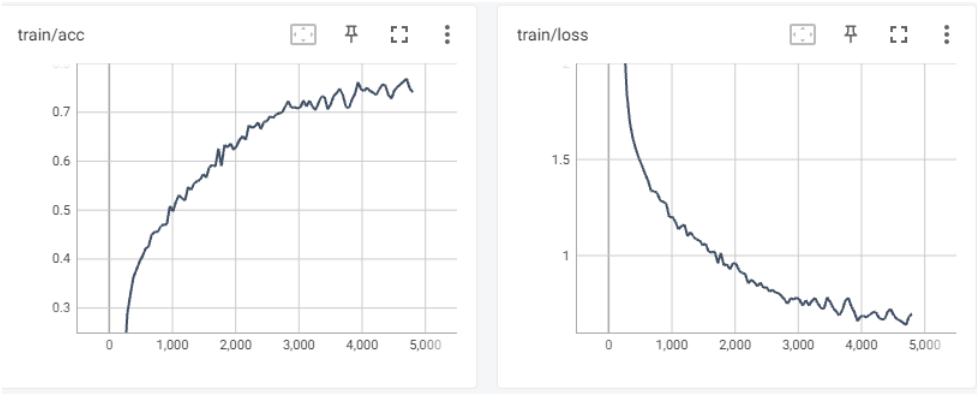

Figura A.1: Fusión a nivel de sensor. Precisión y pérdida en el conjunto de entrenamiento

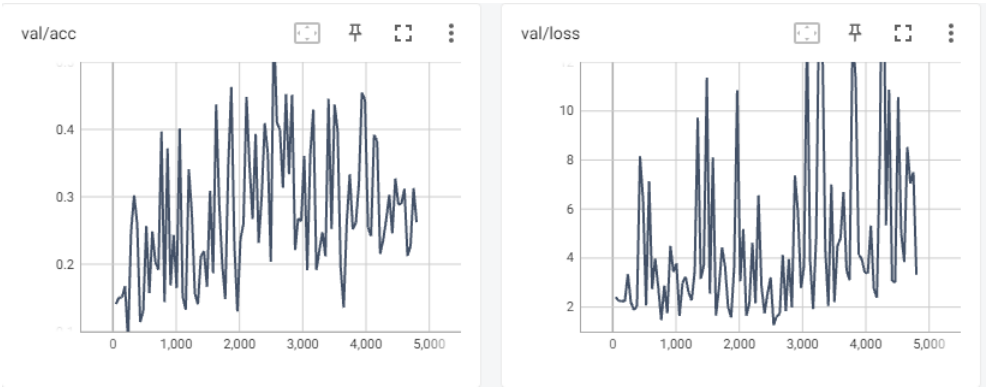

Figura A.2: Fusión a nivel de sensor. Precisión y pérdida en el conjunto de validación

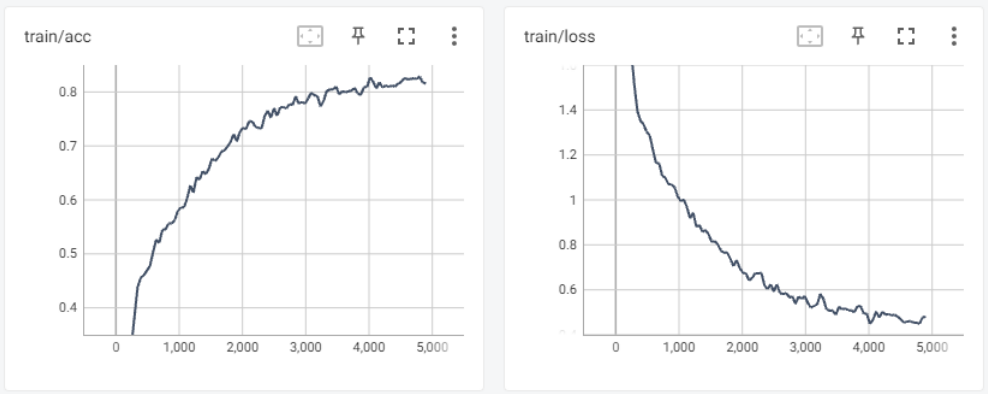

Figura A.3: Fusión a nivel de características, usando 64 canales en el tensor de características. Precisión y pérdida en el conjunto de entrenamiento

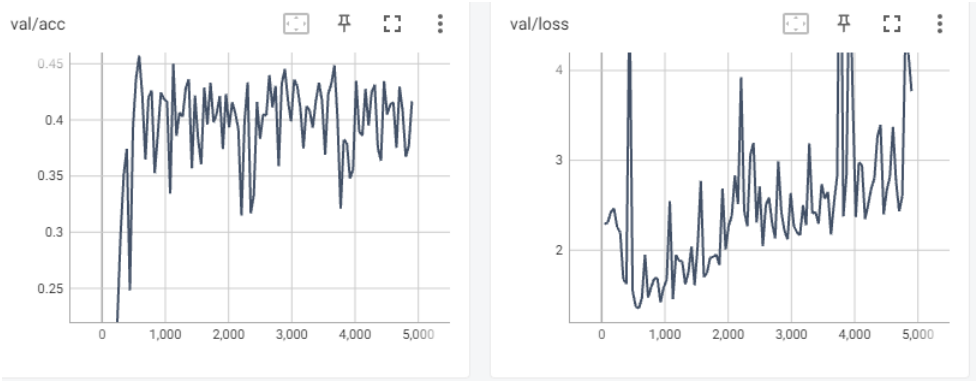

Figura A.4: Fusión a nivel de características, usando 64 canales en el tensor de características. Precisión y pérdida en el conjunto de validación

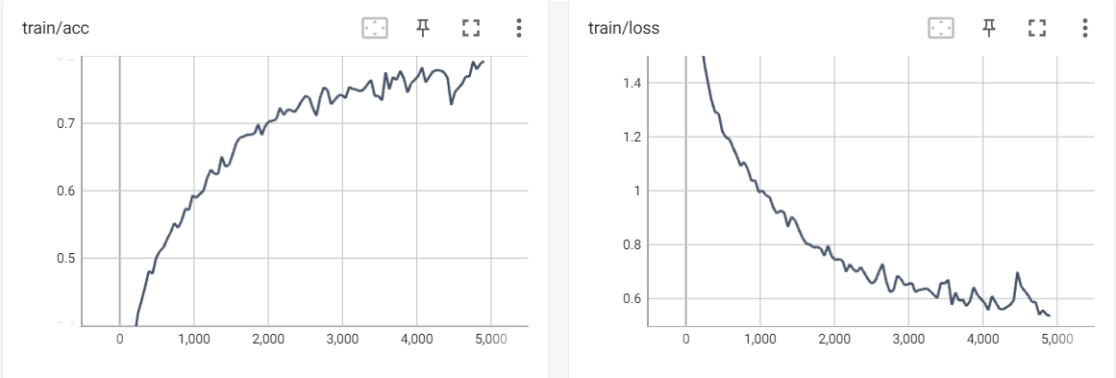

Figura A.5: Fusión a nivel de características, usando 80 canales en el tensor de características. Precisión y pérdida en el conjunto de entrenamiento

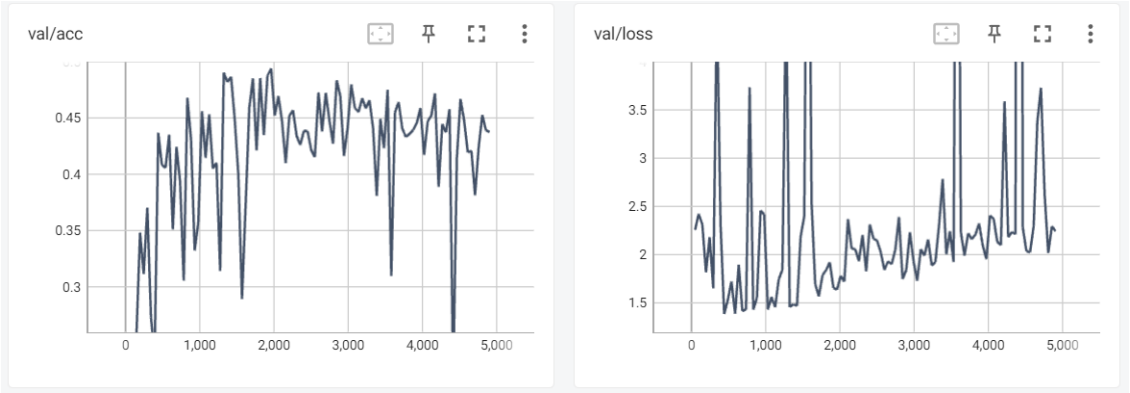

Figura A.6: Fusión a nivel de características, usando 80 canales en el tensor de características. Precisión y pérdida en el conjunto de validación

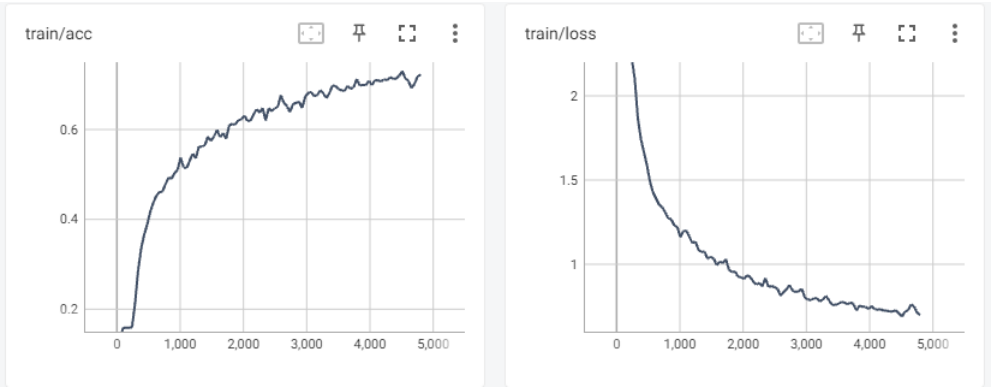

Figura A.7: Clasificador individual con la modalidad IMU. Precisión y pérdida en el conjunto de entrenamiento

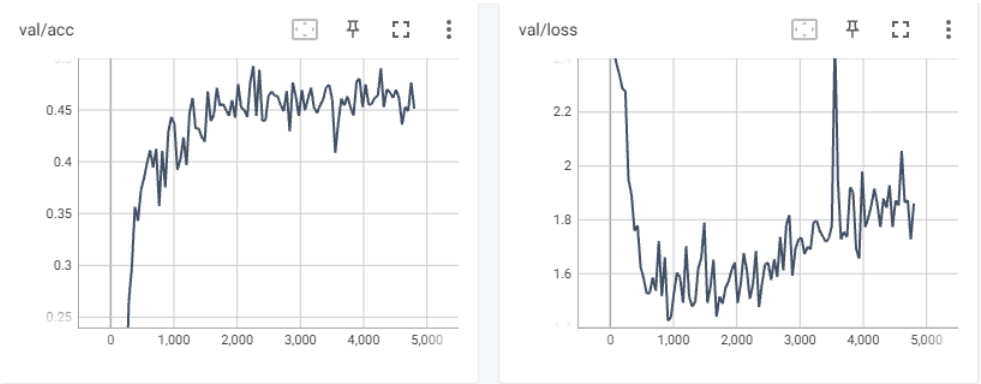

Figura A.8: Clasificador individual con la modalidad IMU. Precisión y pérdida en el conjunto de validación

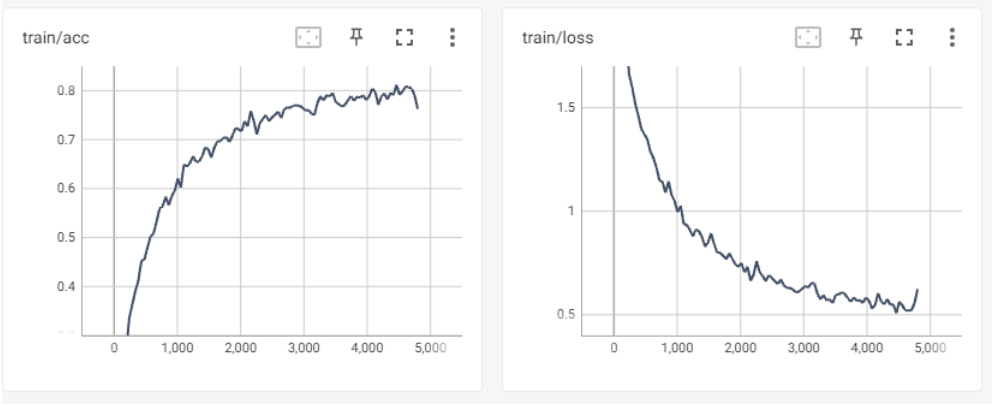

Figura A.9: Clasificador individual con la modalidad de puntos clave de Kinect. Precisión y pérdida en el conjunto de entrenamiento

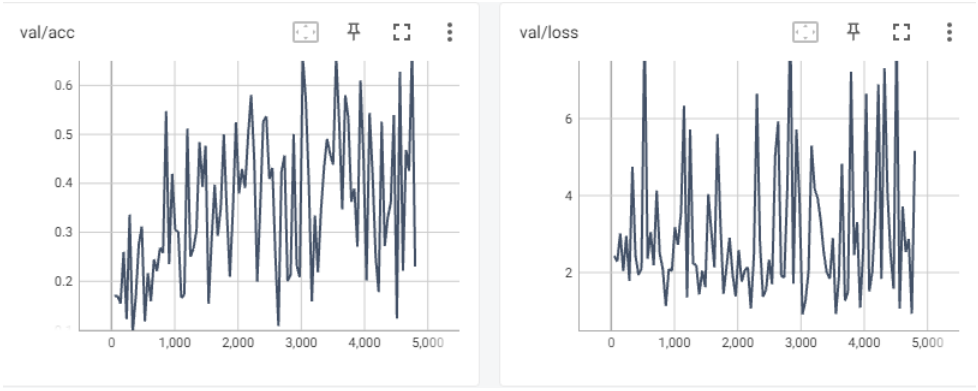

Figura A.10: Clasificador individual con la modalidad de puntos clave de Kinect. Precisión y pérdida en el conjunto de validación

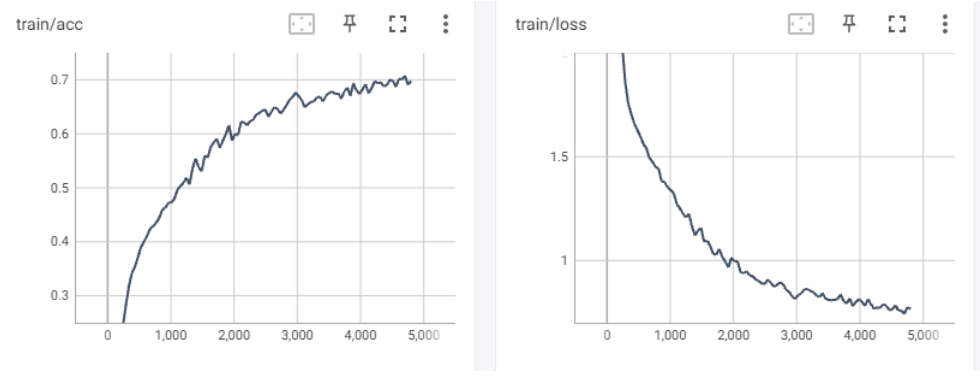

Figura A.11: Clasificador individual con la modalidad E4. Precisión y pérdida en el conjunto de entrenamiento

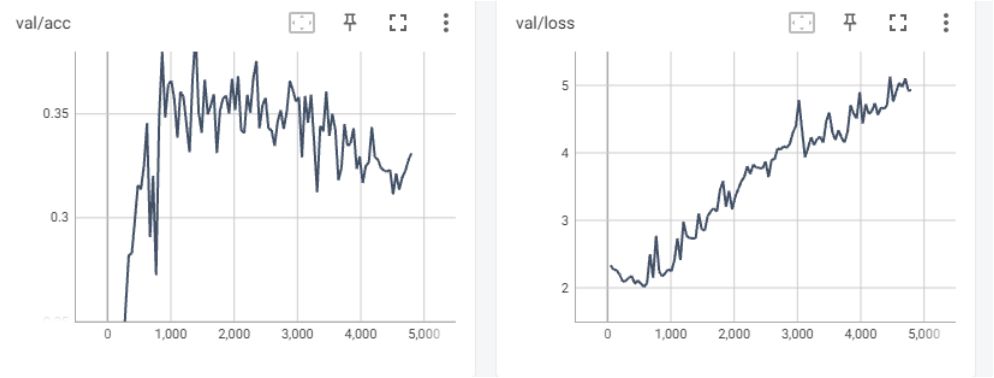

Figura A.12: Clasificador individual con la modalidad E4. Precisión y pérdida en el conjunto de validación

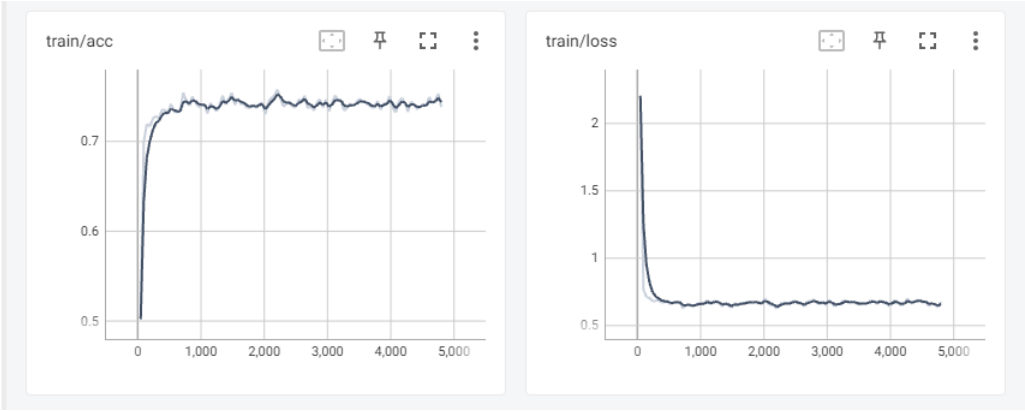

Figura A.13: Combinación de clasificadores individuales en un meta-modelo. Precisión y pérdida en el conjunto de entrenamiento

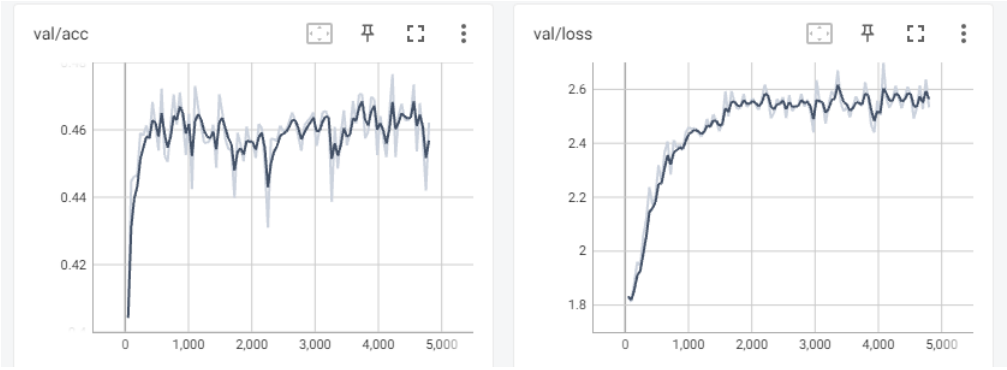

Figura A.14: Combinación de clasificadores individuales en un meta-modelo. Precisión y pérdida en el conjunto de validación

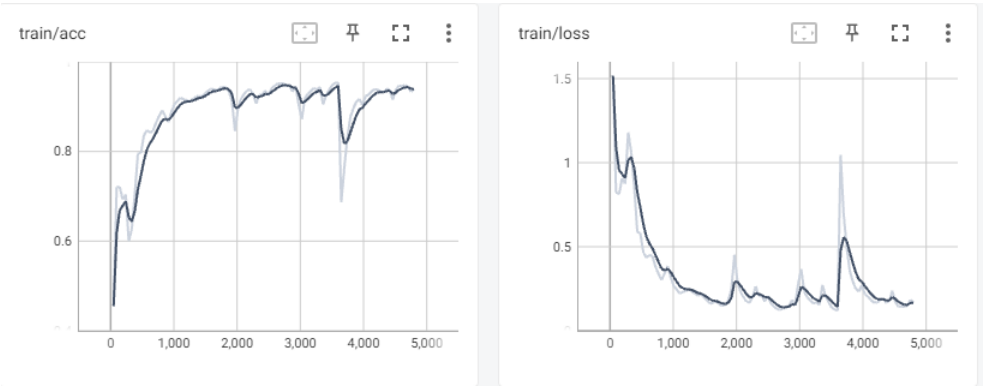

Figura A.15: Clasificador individual con los datos IMU en el modelo competitivo. Precisión y pérdida en el conjunto de entrenamiento

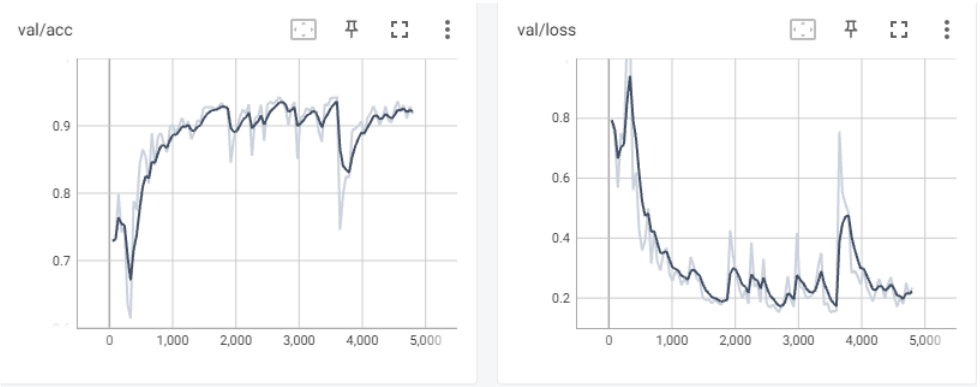

Figura A.16: Clasificador individual con los datos IMU en el modelo competitivo. Precisión y pérdida en el conjunto de validación

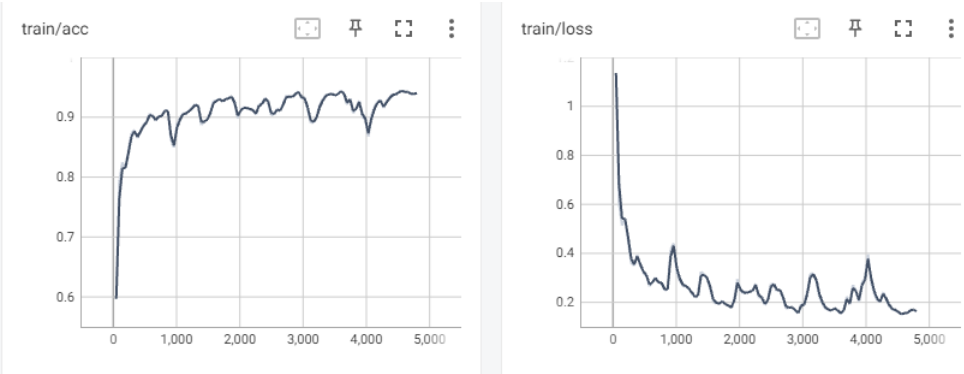

Figura A.17: Clasificador individual con los datos de puntos clave de Kinect en el modelo competitivo. Precisión y pérdida en el conjunto de entrenamiento

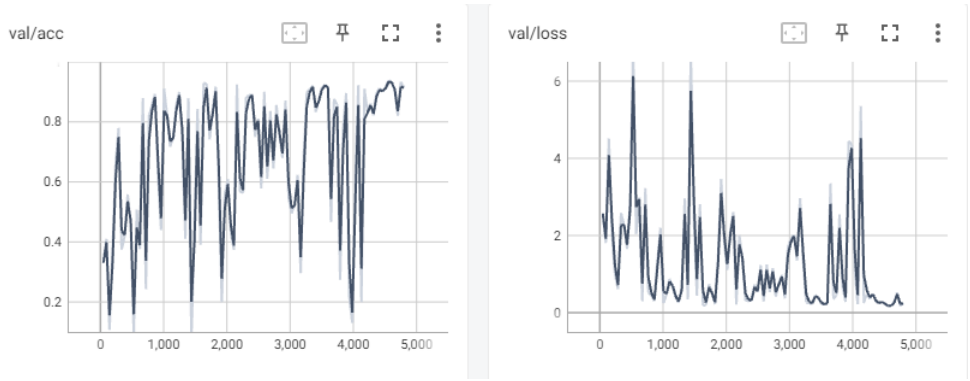

Figura A.18: Clasificador individual con los datos de puntos clave de Kinect en el modelo competitivo. Precisión y pérdida en el conjunto de validación

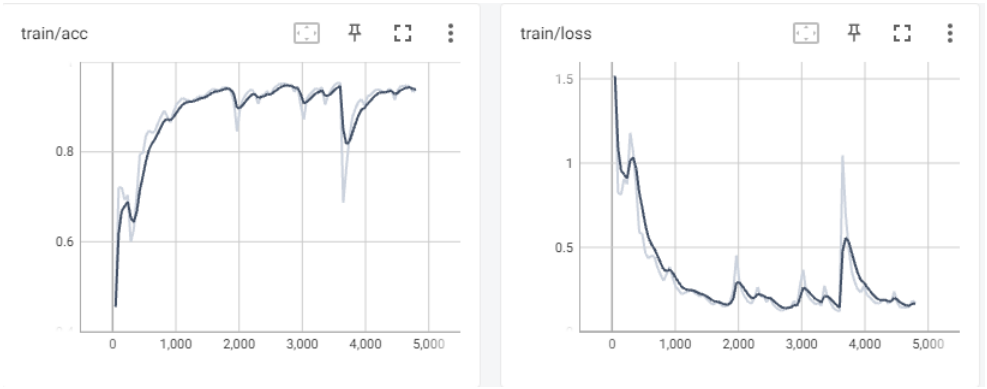

Figura A.19: Clasificador individual con los datos E4 en el modelo competitivo. Precisión y pérdida en el conjunto de entrenamiento

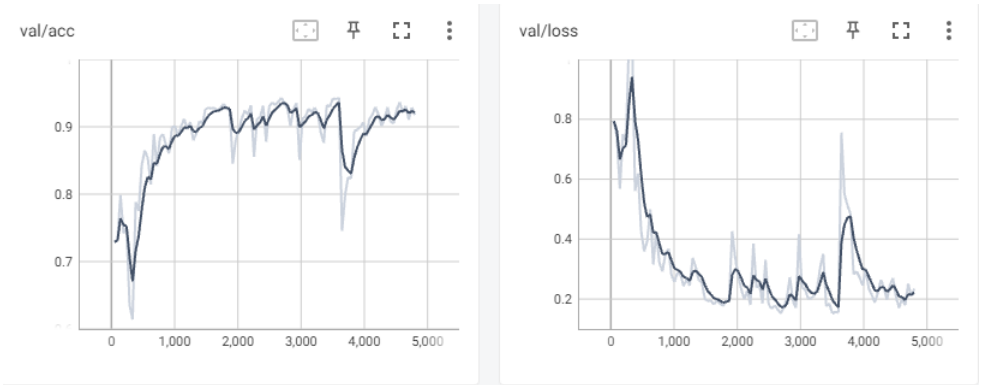

Figura A.20: Clasificador individual con los datos E4 en el modelo competitivo. Precisión y pérdida en el conjunto de validación

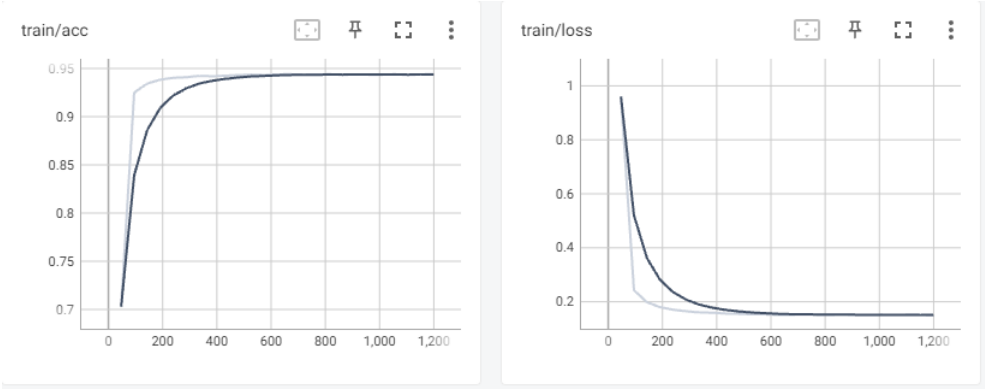

Figura A.21: Modelo final. Precisión y pérdida en el conjunto de entrenamiento

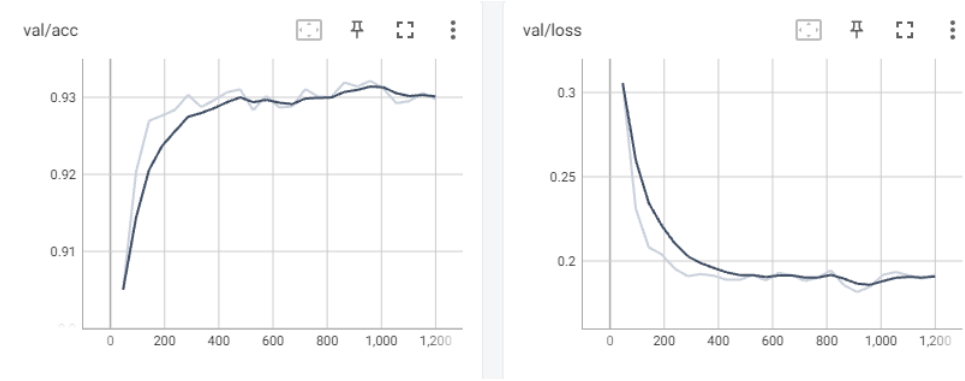

Figura A.22: Modelo final. Precisión y pérdida en el conjunto de validación