Universidad de Valladolid

Máster universitario

Ingeniería Informática

Trabajo Fin de Máster

Identificacion de individuos mediante su interaccion con el dispositivo de apertura de una puerta aplicando tecnicas de Deep Learning

Realizado por **Aitor Ojeda Bilbao**

✠✠✠

Universidad de Valladolid

16 de septiembre de 2021

Tutores: D. César Llamas Bello y D. Jesús M. Vegas Hernández

Resumen

La inteligencia ambiental tiene como objetivo realizar la integración de enfoques inteligentes en diferentes tipos de infraestructuras IoT, principalmente, utilizando objetos de la vida cotidiana de cualquier entorno. La hipótesis principal del trabajo es identificar cuáles son las mejores aproximaciones, para realizar una autenticación de usuarios en el momento en que éstos interactúen con la manilla de una puerta, aplicando técnicas de Deep Learning, más concretamente, redes neuronales. Esta propuesta contribuye a la creación de un método diferente para identificar a las personas de una manera más fluida, y centrada en la privacidad del usuario, para que pueda ser utilizada en cualquier tipo de escenario sin necesidad de ningún dispositivo o sistema adicional. En este caso, utilizamos los datos recopilados en un trabajo anterior, generados por acelerómetros y giroscopios integrados en la manilla de una puerta, donde se obtuvo un conjunto de datos que comprendía muestras de 47 individuos. Se adopta un enfoque centrado en redes neuronales recurrentes, y redes neuronales profundas, extrayendo las características de ambas soluciones, y estableciendo la mejor combinación de los valores que tienen los parámetros. Se ha realizado un estudio de los reultados obtenidos de los diferentes métodos aplicados sobre diferentes conjuntos de datos y características para evaluar el comportamiento de este reto de identificación. Los valores observados de las métricas utilizadas, como la precisión, muestran unos resultados muy interesantes por encima de 0,93 utilizando redes neuronales profundas.

Descriptores

Autenticación, manilla de puerta, IoT, Deep Learning, Redes Neuronales, Redes Neuronales Profundas, Redes Neuronales Recurrentes.

Abstract

Ambient intelligence aims to perform the integration of intelligent approaches in different types of IoT infrastructures, mainly using everyday objects of any environment. The main hypothesis of the work is to identify which are the best approaches, to perform a user authentication at the moment they interact with a door handle, applying Deep Learning techniques, more specifically, neural networks. This proposal contributes to the creation of a different method to identify people in a more fluid way, and focused on user privacy, so that it can be used in any type of scenario without the need for any additional device or system. In this case, we use data collected in a previous work, generated by accelerometers and gyroscopes embedded in a door handle, where a dataset comprising samples of 47 individuals was obtained. An approach focusing on recurrent neural networks, and deep neural networks is adopted, extracting the characteristics of both solutions, and establishing the best combination of the values that the parameters have. A study of the results obtained from the different methods applied on different data sets and features has been carried out to evaluate the performance of this identification challenge. The observed values of the metrics used, such as accuracy, show very interesting results above 0.93 using deep neural networks.

Keywords

authentication, door handle, IoT, Deep Learning, Neural Networks,Deep Neural Networks, Recurrent Neural Networks, Neural Networks

Índice general

Índice de figuras

Índice de tablas

1: Introducción

El impacto de la tecnología en nuestra vida diaria ha ido subiendo exponencialmente a lo largo de los años, surgiendo cada día métodos nuevos, herramientas, modelos, sistemas complejos, etc. En relación con los sistemas de seguridad, también ha supuesto un gran cambio en el día a día, pasando de maneras simples de identificación como pueden ser las contraseñas, pines o los patrones, a modelos más complejos, basados en estructuras de datos más complejas, como pueden ser los sistemas basados en biometría, los escáneres de retina, la tecnología blockchain en el ámbito digital, etc. En esta línea, se ha producido un gran incremento de estos sistemas biométricos para realizar autenticaciones de usuarios ya que incluyen un mayor nivel de seguridad en el proceso para realizar autenticaciones, dejando en un plano más secundario a métodos más tradicionales ya mencionados [1].

La biometría se refiere a la identificación automática de una persona en función de sus características físicas o de comportamiento. Existen 2 tipos de autenticación mediante biometría: la autenticación biométrica explícita y la autenticación biométrica implícita. El primer tipo utiliza características fisiológicas del cuerpo humano, como por ejemplo las huellas dactilares, los iris y las diferentes formas de la cara, para verificar la identidad de un usuario. A pesar de su simplicidad y popularidad, este tipo de recopilación explícita de información biológica genera algunos problemas sobre la privacidad de los usuarios, e incluso puede conducir a la filtración de información personal [2] . Además de este problema, la biometría explícita solo autentica a un usuario al inicio de un dispositivo o servicio, lo que presenta una vulnerabilidad muy significativa para los ataques de seguridad que pueden producirse después de la autenticación inicial [3, 4].

Sin embargo, el otro método de autenticación biométrica, la autenticación implícita, simplemente se basa en autenticar a los usuarios a través de patrones de comportamiento, permitiendo así una autenticación implícita y continua como si se tratase de una función en segundo plano en un dispositivo o servicio. Algunas de las soluciones que puede implentar este tipo de autenticación son el uso de gestos táctiles [4], la dinámica de pulsaciones de teclas [5], patrones de marcha [6] y los comportamientos de uso de dispositivos cotidianos (como una manilla de puerta) por parte de los usuarios [7] .

Estos métodos exhiben atributos de mayor usabilidad y capacidad de aprendizaje para la autenticación de usuarios, dado que la autenticación implícita no requiere un proceso específico, además de que dichos métodos pueden complementar el método de autenticación explícito actual después del inicio de sesión inicial.

Finalmente, este tipo de enfoque es particularmente valioso debido a que existen diferentes dispositivos portátiles que pueden ser incorporados para la autenticación del usuario, y que pueden presentarse en diferentes formas, como relojes inteligentes, guantes o incluso suelas de zapato, que ayudan a reducir la dependencia de la autenticación del usuario en los dispositivos inteligentes, y además, este método permite la autenticación silenciosa del usuario a través del uso diario de objetos de la forma en que el usuario manipula los objetos por naturaleza, siendo un escenario muy bueno para evitar las autenticaciones falsas o los robos de identidad.

1.1. Motivación

La motivación principal del trabajo en el grupo de investigación de la universidad de Valladolid, ha sido estudiar e investigar las formas de realizar un sistema de autenticación de usuarios a través de técnicas basadas en Deep Learning. Para poder realizar la autenticación, se han tomado muestras de diferentes usuarios a través de un sistema que nos permitía obtener datos relevantes acerca de la interacción con el pomo de una puerta por parte de los usuarios, elaborando finalmente un conjunto de datos repartido en diferentes directorios y archivos que formaban los datos del corpus final.

En relación con lo explicado en la sección anterior, este tipo de propuesta podría significar múltiples ventajas en el marco de la autenticación de usuarios ya que, sin la utilización de contraseñas ni datos personales significativos, es decir, datos que permitan identificar a una persona por sus rasgos físicos, como las huellas dactilares o las pupilas de los ojos, seríamos capaces de crear un escenario de autenticación que no pusiera en peligro la privacidad de los usuarios, además de aumentar la complejidad de que el sistema pueda ser atacado o corrompido, debido al uso de patrones de comportamientos en combinación con diferentes herramientas más avanzadas.

Otra de las motivaciones que ha causado la realización de este trabajo, ha sido la propuesta de desarrollar una solución diferente a la que se había realizado en un trabajo previo, ya que dicho trabajo utilizaba métodos y técnicas de Machine Learning más clásicas, y poder realizar desde un contexto diferente, un trabajo similar utilizando técnicas basadas en Deep Learning y redes neuronales.

Además de ello, también la motivación ha sido la elaboración de un modelo correcto de Deep Learning, que nos permita observar la efectividad de las técnicas utilizadas, usando como datos principales los datos del corpus mencionado anteriormente, y comprobar mediante diferentes tipos de métricas y herramientas cómo es su rendimiento, y que ajustes y optimizaciones pueden hacerse para aumentar las características del mismo, y cuáles son los efectos de las variaciones en los valores de los parámetros que contiene dicho modelo.

1.2. Objetivos

El objetivo principal de este trabajo es realizar una investigación sobre los modelos de Deep Learning existentes, que nos permitan explorar el uso potencial de los mismos en un escenario de autenticación de usuario,s a través de la manipulación de un objeto, en este caso, la manilla de una puerta. Junto con este objetivo principal, se han establecido los siguientes objetivos secundarios para la elaboración de este trabajo:

1. Investigación sobre el deep learning y las redes neuronales
2. Adquisición de un mayor conocimiento acerca del funcionamiento de los diferentes modelos de redes neuronales y su comportamiento en un contexto de autenticación de usuarios.
3. Familiarización y exploración de los datos del corpus.
4. Establecimiento de un entorno de trabajo óptimo para poder elaborar todo el trabajo de la manera más cómoda y sencilla.
5. Preparación de los datos para su uso por parte de los algoritmos de las redes neuronales.
6. Elaboración de diferentes modelos de red neuronal utilizando distintas combinaciones de los parámetros.
7. Evaluación de los diferentes modelos de redes neuronales utilizando métricas adecuadas adaptadas al problema planteado.

1.3. Metodología

La metodologıa utilizada para la elaboración del trabajo consiste en ir avanzando poco a poco respecto a los resultados que se van obteniendo a través de búsquedas de artículos e información acerca del problema planteado. Este método se divide en diferentes etapas, en las que se itera de manera progresiva hasta alcanzar una solución aceptable, repitiéndose las diferentes etapas conforme al número de diferentes soluciones propuestas para el trabajo:

- Observación de las soluciones existentes relacionadas con el problema a tratar
- Desarrollo de la solución propuesta
- Análisis y evaluación de la solución propuesta

En la etapa de observación, se ha realizado una búsqueda de las soluciones relacionadas con el problema que se expone en el trabajo, se ha seguido un proceso de análisis bibliográfico (Capítulo 2) en el que se han estudiado las diferentes soluciones y extraído información relacionada con el objetivo principal de este trabajo. Por otro lado, en la parte de desarrollo de la solución (Capítulos 3, 4 y 5), se ha partido de la información extraída en la observación y análisis de soluciones existentes, identificando las mejores soluciones que pueden tomarse en el desarrollo del trabajo, con las técnicas correspondientes para su realización. Con dicha información extraída, se ha desarrollado la solución. Por último, la etapa de análisis y evaluación (Capítulo 5) ha consistido en la comprobación de la validez de la solución desarrollada, donde se han estudiado las modificaciones y optimizaciones que permitieran la mejora de la misma además de su implementación, y finalmente, onteniendo la evaluación de su eficacia de diferentes formas, y analizando los puntos fuertes y débiles de dicha evaluación.

1.4. Estructura del documento

La memoria del trabajo se va a dividir en diferentes capítulos donde se hablarán de todos los aspectos necesarios para la comprensión y el entendimiento de la solución propuesta.

- ***Capítulo 2: Estado del arte.*** En este capítulo se hablará de los trabajos relacionados además de la teoría necesaria para poder comprender el desarrollo de este trabajo, describiendo de manera detallada las diferentes soluciones estudiadas, y los diferentes aspectos teóricos sobre el Deep Learning a tener en cuenta.
- ***Capítulo 3: Técnicas y herramientas.***En este capítulo se llevará a cabo una descripción de las técnicas y de las herramientas utilizadas en el desarrollo del trabajo.

- ***Capítulo 5: Desarrollo del trabajo de investigación.*** Hablaremos sobre diferentes aspectos relacionados con el desarrollo del trabajo, como la puesta en marcha del entorno de trabajo, las primeras soluciones adoptadas, la optimización, estudio y mejora de las soluciones, y finalmente, la evaluación y análisis de las soluciones finales.
- ***Capítulo 6: Conclusiones y trabajo futuro.*** Por último, en este capítulo podremos encontrar las conclusiones que se han extraído acerca de la realización del trabajo, y de los diferentes aspectos que pueden resultar interesantes continuar con ellos en el futuro.

Este documento también posee otra serie de secciones que no por ello son menos importantes:

- **Bibliografía.**
- **Anexo A. Planificación del proyecto.** Apartado en el que se explica de forma general como se ha realizado la planificación del trabajo realizado.
- **Anexo B. Documentación de usuario** Manual de cómo utilizar el trabajo desarrollado.

2: Estado del arte

En esta capítulo, abordaremos todos los trabajos similares con el presente estudio, además de explicar los conceptos y técnicas fundamentales para la comprensión del trabajo realizado.

2.1. Trabajos relacionados

Para poder abordar la realización de un trabajo de este tipo, debemos de realizar tareas propias de investigación acerca de trabajos similares para que puedan aportar un valor relevante a la solución del problema que estamos planteando. En este caso, podemos distinguir 3 enfoques diferentes antes de continuar con el problema: la autenticación de usuarios mediante sistemas biométricos, la autenticación de usuarios mediante técnicas basadas en deep learning, y la autenticación de usuarios a través de la interacción con una puerta.

Autenticación de usuarios mediante sistemas biométricos

La información biométrica de las personas, como el iris la voz, la cara, la huella dactilar o incluso la forma de andar, puede ser una forma rápida y segura de realizar una autenticación de usuarios. El uso de la autenticación biométrica se ha convertido en algo bastante común para muchas personas hoy en día, surgiendo diferentes métodos y alternativas diferentes para este problema. Romanowski et al. investigaron la aceptabilidad y la facilidad de uso de un escáner de venas de la palma de la mano en 2016 [8]. En su estudio, el 75 % de los 55 participantes consideró que la tecnología no era intrusiva y el 77 % no experimentó ningún retraso durante la autenticación. Con este dispositivo, los usuarios se benefician de la forma física de agarrar la manilla de la puerta, ya que es intuitiva gracias a la adaptación física de la misma.

Por otro lado, la empresa Fujitsu, como creadora de escáneres de las venas de la palma de la mano en el mercado, anunció en 2018 que reemplazaría las contraseñas y las tarjetas inteligentes para 80.000 empleados en su sede japonesa por su escáner de venas de la palma PalmSecure [9]. Con estos esfuerzos que muestran un alto potencial, establecemos a la autenticación biométrica de las venas de la palma de la mano como una solución relacionada con el problema planteado, y que nos sirva como planteamiento para poder abordar el objetivo principal de este trabajo.

Siguiendo en la misma línea, actualmente, los dispositivos que se han convertido en los grandes protagonistas para poder realizar autenticaciones de usuario basadas en biometría, sin necesidad de realizar una acción específica de autenticación, son los dispositivos portátiles. Éstos dispositivos, como son las pulseras de actividad, los relojes inteligentes, o incluso gafas inteligentes, son considerados dispositivos que permiten realizar una conexión a internet, e intercambiar datos del usuario que lo lleva puesto sin necesidad de la interacción humana. En el estudio realizado por *Kuo-Hui Yeh y Chunhua Su* [10] demuestran que con los dispositivos portátiles, se pueden tomar una serie de diferentes datos biométricos muy relevantes que permiten realizar una autenticación de usuarios. Concretamente, su estudio se basa en la autenticación continua, es decir, a través de los datos que van recopilando en tiempo real, haciendo que el sistema aprenda continuamente acerca del usuario que lleva puesto el dispositivo. Lo han llevado a cabo analizando la forma de andar del usuario, utilizando un sistema especial que recopila información acerca de la presión ejercida por el usuario a la hora de dar un paso, utilizando unas zapatillas especializadas para dicho fin.

Los resultados que obtuvieron en el estudio fueron bastante buenos, descubriendo que la técnica utilizada superaba otros estudios de autenticación mediante datos biométricos, como el iris y los movimientos del ojo [11], el oido [12], o la cara [13]. Según el experimento, el índice de la precisión de la verificación de entidades en el sistema de autenticación propuesto tiene un valor medio comprendido en un rango del 83,88 % y del 99,60 %, aplicando 14 muestras diferentes mediante 2 clasificadores que son Naive Bayes y SVM con Gaussian radial basis function (SVM-GRBF) respectivamente. Por tanto, concluyen que los datos biométricos estudiados son muy relevantes a la hora de realizar una autenticación mediante dispositivos cotidianos, como puede ser una zapatilla.

Autenticación de usuarios mediante técnicas basadas en deep learning

Para proponer una solución correcta al problema estudiado, debemos de tener en cuenta todos los enfoques posibles acerca del problema, y uno de ellos se basa en la utilización del Deep Learning para la autenticación de usuarios.Los últimos avances tecnológicos realizados han demostrado la buena eficacia de los modelos de aprendizaje profundo para la biometría móvil. En el caso del trabajo realizado por N. Reddy y A. Rattani [14] realizan una evaluación y una comparación sobre los diferentes modelos de aprendizaje profundo en dos términos distintos: eficiencia y precisión, cuando se utilizan para la autenticación biométrica de usuarios en dispositivos móviles.

Para ello, se han utilizado arquitecturas de modelos de deep learning pre-entrenadas como son VGG, ResNet, DenseNet y modelos propuestos para aplicaciones de visión móvil, como MobileNetV1, MobileNetV2 y NasNet-mobile. Estos modelos se han comparado con respecto a su rendimiento y coste computacional. Además de estos modelos, han propuesto un modelo de aprendizaje profundo compacto y rápido para operaciones eficientes en dispositivos móviles con recursos limitados, realizando investigaciones con el fin de adaptarlo lo máximo posible al conjunto de datos.

Los resultados principales obtenidos presentaban que ResNet-50, DenseNet-50 y los modelos personalizados tuvieron un rendimiento mejor en todos los experimentos. Sin embargo, el modelo personalizado propuesto es 35 veces más pequeño y tiene 15,6 veces menos operaciones en comparación con el modelo ResNet-50 y, por tanto, ofrece la mejor relación entre rendimiento y coste computacional, dentro de los límites del estudio y del caso de uso realizados. Además, el modelo propuesto superó a la mayoría de las arquitecturas compactas centradas en los móviles, como MobileNet-V1 y MobileNet-V2, en la mayoría de los experimentos realizados. La conclusión obtenida es que, para poder realizar un modelo de deep learning más eficiente, es necesario adaptarlo al problema estudiado, en vez de utilizar modelos prediseñados que pueden dar resultados menos satisfactorios.

Por otro lado, podemos destacar las redes neuronales como método de autenticación, una práctica que se ha ido extendiendo actualmente sobre los sistemas basados en biometría, y que resulta una herramienta muy práctica y fiable para dicho fin. En este contexto, en el estudio realizado por Ilias Chamatidis y Aggeliki Katsika [15], plantean de nuevo una comparación de métodos de Machine Learning y Deep Learning, llevando a cabo uno de los enfoques más recientes de la autenticación biométrica, como es el uso de electrocardiogramas (ECG), ya que están estrechamente relacionados con las características únicas del corazón de cada persona.

En dicho trabajo se presenta un marco para la autenticación eficiente y utilizable para la autenticación de usuarios, basado en el ECG. El ECG se preprocesa para eliminar cualquier ruido o distorsión y luego se extrae un conjunto múltiple de características, a través de varias transformaciones. Este conjunto de características se utiliza como entrada para los modelos de clasificación y los resultados se comparan para encontrar la combinación transformación-clasificador más eficaz. Además, utilizan redes neuronales para crear los modelos de clasificación que predicen si un ECG pertenece a una persona específica o no, basándose en la combinación de los conjuntos de características extraídos. Los resultados que obtuvieron son diversos, donde recalcan que los modelos de Machine Learning para el conjunto de datos estudiado presentan unas mejores métricas, como el área bajo la curva, comprendiendo resultados entre 81 % y 95 %, mientras que los modelos de redes neuronales profundas no alcanzan una precisión más alta del 80 %.

Estos resultados demuestran que los modelos de aprendizaje profundo necesitan una mayor cantidad de datos para proporcionar una precisión más elevada en comparación con la mayoría de las otras técnicas utilizadas, si hay suficientes datos para entrenarlas. Es por ello que debemos de tener en cuenta esta conclusión en el presente trabajo, y elaborar la mejor solución en base a esta información y la anteriormente estudiada.

Autenticación de usuarios a través de sistemas conectados a una puerta

Continuando con los diferentes enfoques, existen pocos artículos acerca de la autenticación de usuarios a través del mecanismo de una puerta. Fujinami [16] utiliza acelerómetros combinados, unidos a una puerta y a la muñeca de un usuario para identificar a la persona que abre la puerta. Esta identificación se realiza en el momento en el que usuario realiza 3 funciones principales: abrir la puerta, sentarse en la silla, y ponerse a escribir. Una red neuronal recibió las tres señales del acelerómetro como entradas, obteniendo una precisión global del reconocimiento, es decir, el rendimiento de la segunda capa en el diagrama de flujo de identificación, del 96 %. En el contexto de identificación del usuario que abrió la puerta, se identificó correctamente en todos los casos. La identificación del usuario de la silla se reconocía correctamente si sólo había una persona sentada, por lo que no era adecuado para múltiples usuarios utilizando el mismo patrón de comportamiento.

Sin embargo, en una investigación más profunda, Piltaver [17] describe un enfoque diferente que permite reconocer a una persona que entra dentro de una habitación, utilizando sólo los valores de la aceleración de la puerta. El enfoque analiza la señal de la aceleración en 2 ámbitos: el tiempo y la frecuencia.

Para cada ámbito se desarrollaron dos tipos de métodos: basado en características, que utiliza características para describir la aceleración y a continuación utiliza el método de clasificación para identificar a la persona; y basado en señales, que utiliza la señal de aceleración como entrada y encuentra las más similares para identificar a la persona. Estos dos métodos obtuvieron unos resultados en torno al 90 % de precisión, confirmando que es posible identificar a una persona que entra en una habitación utilizando sólo la aceleración de la puerta.

El problema fundamental es que, en estos dos casos, la identificación del usuario se realiza una vez que la persona ya ha entrado en la habitación, por lo que estos enfoques no son adecuados para ser utilizados en un sistema de autenticación o control de acceso. Debemos centrarnos en un método que permita la autenticación temprana de los usuarios, es decir, antes de que se abra la puerta, debemos centrarnos en la interacción con la manija de la puerta, y elaborar un escenario que sea capaz de recoger datos relevantes acerca de ello.

Autenticación de usuarios a través de la interacción con el mecanismo de apertura de una puerta

J.Vegas y C.Llamas [7], realizaron una investigación acerca de si la interacción con una manilla de una puerta, era suficiente para realizar una autenticación de usuarios, mediante técnicas más clásicas de Machine Learning. La plataforma de adquisición de datos estaba formada por un módulo MEMS, compuesto por un acelerómetro y un giroscopio, recopilando información a través de un sensor 6-DOF, que permitía que los datos estuvieran disponibles a través de una red inalámbrica.

Este problema fue planteado sin que el usuario hubiera entrado del todo en la estancia, si no que, en el momento en el que interacciona con la manilla de la puerta, pueda producirse la autenticación. La hipótesis principal planteada fue cierta, descubiriéndose mediante diferentes tipos de métricas y métodos, que realizando una clasificación con diferentes tipos de algoritmos de Machine Learning, se obtenía un valor medio de área bajo la curva (AUC) de un 85 %, siendo un valor más que suficiente para que la hipótesis fuera cierta.

Gracias a la información recabada en el artículo anterior, podemos afirmar que mediante la identificación adecuada de patrones de comportamiento, es posible realizar autenticaciones de usuario, por tanto, para demostrar de una mejor forma esto, en este trabajo se plantea una solución alternativa, realizar una autenticación de usuarios implementando métodos basados en Deep Learning.

Éstos métodos extraen complejas características del comportamiento a través de herramientas como acelerómetros y giroscopios [18]. Utilizan redes neuronales con múltiples capas de operaciones para aprender las características de los datos [19], ya que el Deep Learning tiene la capacidad de aprender automáticamente la representación de los datos más destacada, sin necesidad de una ingeniería de características manual, por lo que es un enfoque muy apropiado para tratar datos de los sensores[20].

2.2. Conceptos teóricos

Una vez hemos visto los diferentes enfoques y métodos en la autenticación de usuarios, además de los puntos de partida para la realización de este trabajo, debemos de hacer una pequeña explicación acerca de las técnicas que se llevarán a cabo en este estudio, para continuar con la resolución del problema planteado.

Machine Learning

El Machine Learning (aprendizaje automático) es un subcampo de las ciencias de la computación y una rama de la inteligencia artificial, cuyo objetivo es desarrollar técnicas que permitan que las computadoras aprendan por sí solas. Un sistema aprenderá a medida de que su desempeño mejora con la experiencia, y mediante el uso de diferentes tipos de datos, es decir, cuando estas habilidades no estaban presentes en el primer desarrollo del sistema. Los pasos que siguen este tipo de modelos son principalmente 3: realizar una observación de datos, construir un modelo basado en esos datos, y finalmente utilizar ese modelo a la vez como una hipótesis acerca del tema a tratar y una pieza de software que pueda resolver problemas. En la Figura 2.1 podemos observar los campos que abarca la inteligencia artificial, siendo el machine learning uno de ellos, y a su vez, dentro de él, podemos encontrar el Deep Learning (aprendizaje profundo), que estudiaremos más adelante y será la base principal de la realización de este trabajo.

Por otro lado, el Machine Learning también está muy relacionado con el reconocimiento de patrones de comportamiento. El Machine Learning puede ser visto además como una técnica para automatizar algunas partes de los métodos científicos mediante métodos matemáticos. Por lo tanto es un proceso de inducción del conocimiento, es decir, una forma de razonamiento en que la verdad de las premisas apoyan la conclusión, pero no la garantizan. Este factor fundamental será muy importante para la realización de este trabajo, ya que el problea fundamental estudiado es el análisis de un patrón de comportamiento. Finalmente, el Machine Learning tiene una amplia gama de aplicaciones en diferentes ámbitos (educación, sanidad, industria, ocio, etc), entre las que podemos destacar motores de búsqueda, diagnósticos médicos, detección de fraude en el uso de tarjetas de crédito, recomendaciones de películas/series en base a lo que ha visto el usuario, análisis del mercado de valores, clasificación de secuencias de ADN, reconocimiento del habla y del lenguaje escrito, etc.

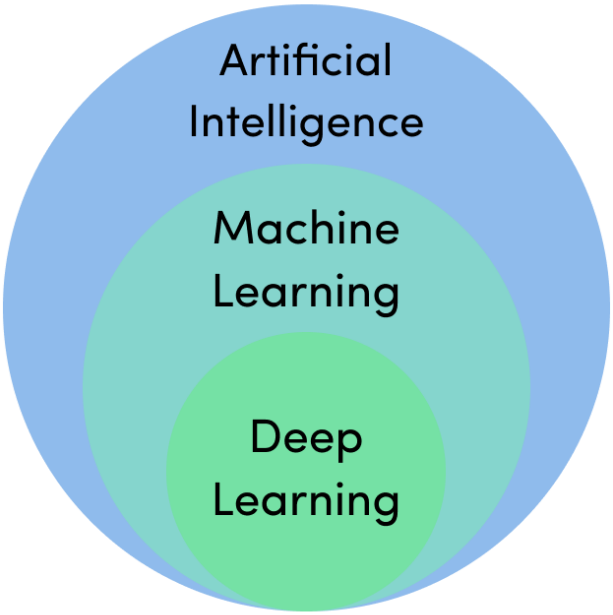

Figura 2.1: Ramas que forman la inteligencia artificial (IA).

Recuperado de *https://levity.ai/blog/difference-machine-learning-deep-learning*

Deep Learning

El Deep Learning (aprendizaje profundo) es un subcampo del aprendizaje automático, como ya habíamos comentado, que se ocupa de los algoritmos inspirados en la estructura y el funcionamiento del cerebro llamados redes neuronales artificiales. El aprendizaje puede ser de 3 tipos: supervisado, semi-supervisado y no supervisado, aun que el modelo más utilizado y más común es el aprendizaje supervisado. Las redes neuronales artificiales (RNA) se basan en el procesamiento de la información y los nodos de comunicación distribuidos en los sistemas biológicos que conocemos. Estas redes tienen varias diferencias con los cerebros biológicos, en concreto, las redes neuronales tienden a ser estáticas y simbólicas, mientras que el cerebro biológico de la mayoría de los organismos vivos es dinámico (plástico) y analógico [21].

Cuando nos referimos a aprendizaje profundo, se refiere al uso de múltiples capas en la red neuronal, una variante moderna que se ocupa de un número ilimitado de capas de tamaño acotado, lo que permite una aplicación práctica y una implementación optimizada, al tiempo que conserva la universalidad teórica en diferentes condiciones y situaciones. También es posible que dichas capas puedan ser heterógeneas, y que puedan desviarse en gran medida de los modelos de conexiones biológicamente estudiados, siempre teniendo en cuenta la eficiencia, la capacidad de entrenamiento y la comprensibilidad, que forman los 3 pilares fundamentales de este campo.

Para poder ilustrar mejor este funcionamiento, en la Figura 2.2 podemos observar una comparación entre los 2 métodos explicados, el machine learning y el deep learning, destacando la principal diferencia entre ambos. Mientras que en el machine learning una persona se encarga de realizar la extracción de características y patrones, y posteriormente el modelo realiza una clasificación con dicha información extraída , el deep learning lo hace automáticamente, aprendiendo por su propio pie el reconocimiento de dichos patrones y las características que presentan los datos sin necesidad de la interacción humana, junto con la creación de los modelos de clasificación, realizando un aprendizaje continuo desde la inclusión de los datos. El único punto que debe de realizar una persona es la adaptación de los datos de entrada para que puedan usarse correctamente en los modelos que se vayan a utilizar.

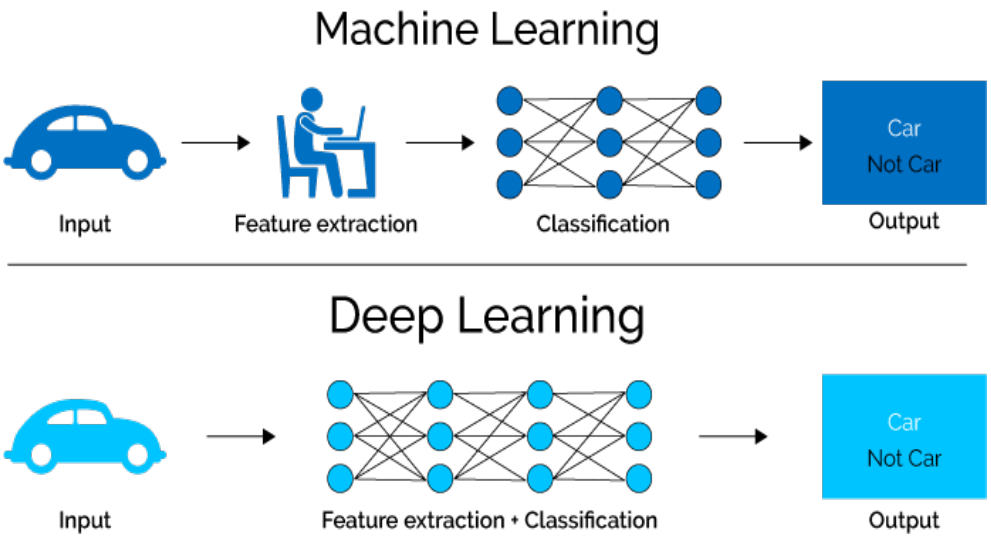

Figura 2.2: Comparación general del funcionamiento del Machine Learning y del Deep Learning.

Recuperado de *https://towardsdatascience.com/why-deep-learning-is-needed-over-traditional-machine-learning-1b6a99177063*

Existen diferentes arquitecturas de Deep Learning, entre las que podemos destacar varias como las redes neuronales profundas, las redes de creencias profundas, el aprendizaje de refuerzo profundo, las redes neuronales recurrentes y las redes neuronales convolucionales. Estos tipos de arquitecturas se han aplicado en diferentes ámbitos como la visión por ordenador, el reconocimiento del habla, el procesamiento del lenguaje natural, la traducción automática, la bioinformática, el diseño de fármacos y el análisis de imágenes médicas, entre otros, en los que han producido resultados comparables y, en algunos casos, superiores al rendimiento producido por los expertos humanos [22]. Más concretamente, en este trabajo haremos uso de los dos modelos más conocidos: las redes neuronales profundas y las redes neuronales recurrentes, ya que el problema a tratar se adapta a las descripciones de uso de ambas y podremos dar un enfoque diferente, analizando los resultados del rendimiento de ambas redes neuronales. En la siguiente sección explicaremos un poco más en detalle cada una de ellas, para ilustrar su funcionamiento y el por qué de su utilización.

Redes neuronales profundas

Las redes neuronales clásicas o profundas (Deep Neural Networks), también conocidas como redes neuronales totalmente conectadas, se identifican principalmente por sus perceptrones multicapa, en los que las neuronas están conectadas a la capa continua, existiendo múltiples capas entre la entrada y la salida. Existen diferentes tipos, pero todos tienen la misma serie de componentes: neuronas, sinapsis, pesos, sesgos y funciones. Estos componentes permiten funcionar a los modelos de manera similar al cerebro humano, y se pueden entrenar como cualquiero otro algoritmo de Machine Learning.

Las DNN suelen ser redes de avance en las que los datos pasan de la capa de entrada a la de salida sin hacer un bucle de retorno, es decir, volver a utilizar los valores anteriores. En primer lugar, la DNN crea un mapa de neuronas virtuales y asigna valores numéricos aleatorios, llamados “pesos”, a las conexiones entre ellas. Los pesos y las entradas se multiplican y devuelven una salida entre 0 y 1. Si la red no reconociera con precisión un patrón concreto, un algoritmo ajustaría los pesos. Así, el algoritmo puede hacer que ciertos parámetros sean más significativos, hasta que determine la manipulación matemática correcta para procesar completamente los datos.

Para ilustrar de una mejor forma este tipo de funcionamiento, en la Figura 2.3 podemos observar el esquema de capas de una red neuronal profunda, y cómo la información va pasando por todas ellas hasta que se genera una salida final.

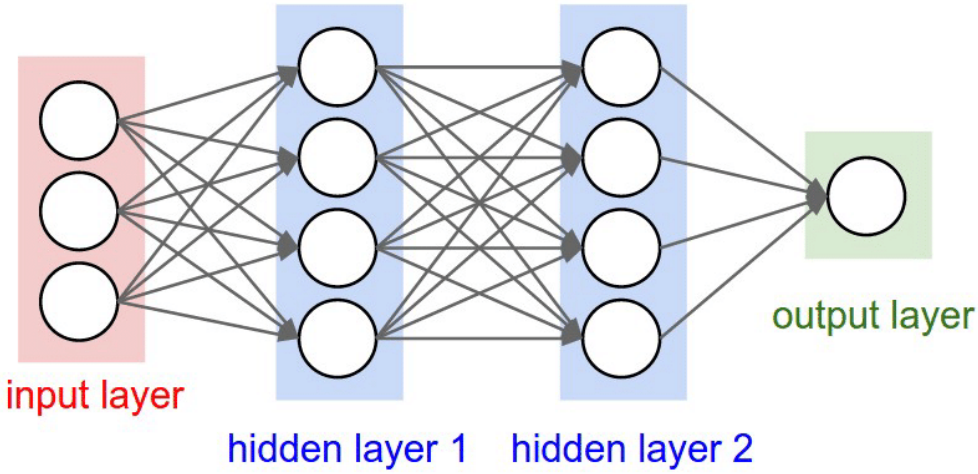

Figura 2.3: Esquema por capas de una red neuronal.

Recuperado de *https://www.researchgate.net/figure/A-general-model-of-a-deep-neural-network-It-consists-of-an-input-layer-some-here-two_fig1_308414212*

Las 2 principales funciones que utilizan este tipo de red neuronal son la función lineal, que representa una sola línea que multiplica sus entradas por un multiplicador constante; y la función no lineal, que está más extendida, y se puede dividir en 3 tipos:

- **Curva Sigmoide**. Función interpretada como una curva en forma de S y con un rango de 0 a 1.
- **Tangente hiperbólica**. Hace referencia también a la curva en forma de S, pero con un rango de -1 a 1.
- **Unidad Lineal Rectificada (ReLU)**. Es una función de un solo punto que se posiciona en 0 cuando el valor de entrada es menor que el valor establecido y aumenta el rendimiento si el múltiplo lineal de la entrada dada es mayor que el valor establecido. Esta función es la más conocida, y es específicamente buena para tratar problemas de clasificación, como podremos comprobar más adelante.

Este tipo de red neuronal funciona muy bien para realizar tareas de clasificación y regresión de datos, por lo que lo hace un enfoque muy adecuado para el problema que estamos planteando en este trabajo.

Redes neuronales recurrentes

Las redes neuronales recurrentes (RNN) son otro tipo de técnica de Deep Learning, perteneciente a la clase de redes neuronales artificiales, en las que las conexiones entre los nodos forman una malla o grafo dirigido a lo largo de una secuencia temporal. Gracias a esto, le permite mostrar un comportamiento dinámico temporal, y procesar diferentes secuencias de entrada de longitud variable. Las RNN utilizan el conocimiento obtenido de su estado anterior como valor de entrada para la predicción actual, por lo tanto, puede ayudar a conseguir una memoria a corto plazo en una red, lo que permite su utilización en sistemas de datos basados en el tiempo, siendo capaz no sólo de procesar puntos de datos individuales, si no también secuencias enteras de datos, como el habla [23] o el vídeo [24].Dentro de este tipo de redes neuronales, podemos destacar los 2 tipos más significativos:

- Long Sort-Term Memory (LSTM). Este tipo de red es muy útil a la hora de realizar una predicción de datos en secuencias temporales, utilizando la memoria. Una unidad LSTM común se compone de una célula, una puerta de entrada, una puerta de salida y una puerta de olvido. La célula recuerda valores a lo largo de intervalos de tiempo arbitrarios y las tres puertas regulan el flujo de información que entra y sale de la célula. Es por ello que estas series son idóneas para clasificar, procesar y hacer predicciones a partir de datos de series temporales, ya que puede haber desfases de duración desconocida entre los eventos importantes de una serie temporal. En la Figura 2.4 se muestra este tipo de funcionamiento, ilustrando como se procesa la informacion y que pasos se siguen dentro de una célula LSTM.
- RNNs con puerta de entrada (GRU). Este tipo de RNN es parecido al enfoque de memoria a corto plazo (LSTM), aun que implementa menos parámetros que el modelo LSTM ya que carece de una puerta de salida. El rendimiento de la GRU en ciertas tareas, como el procesamiento del lenguaje natural resultó ser similar al de la LSTM [25]. Sin embargo, las GRU sí que han demostrado un mejor rendimiento en ciertos conjuntos de datos más pequeños y menos frecuentes [26].

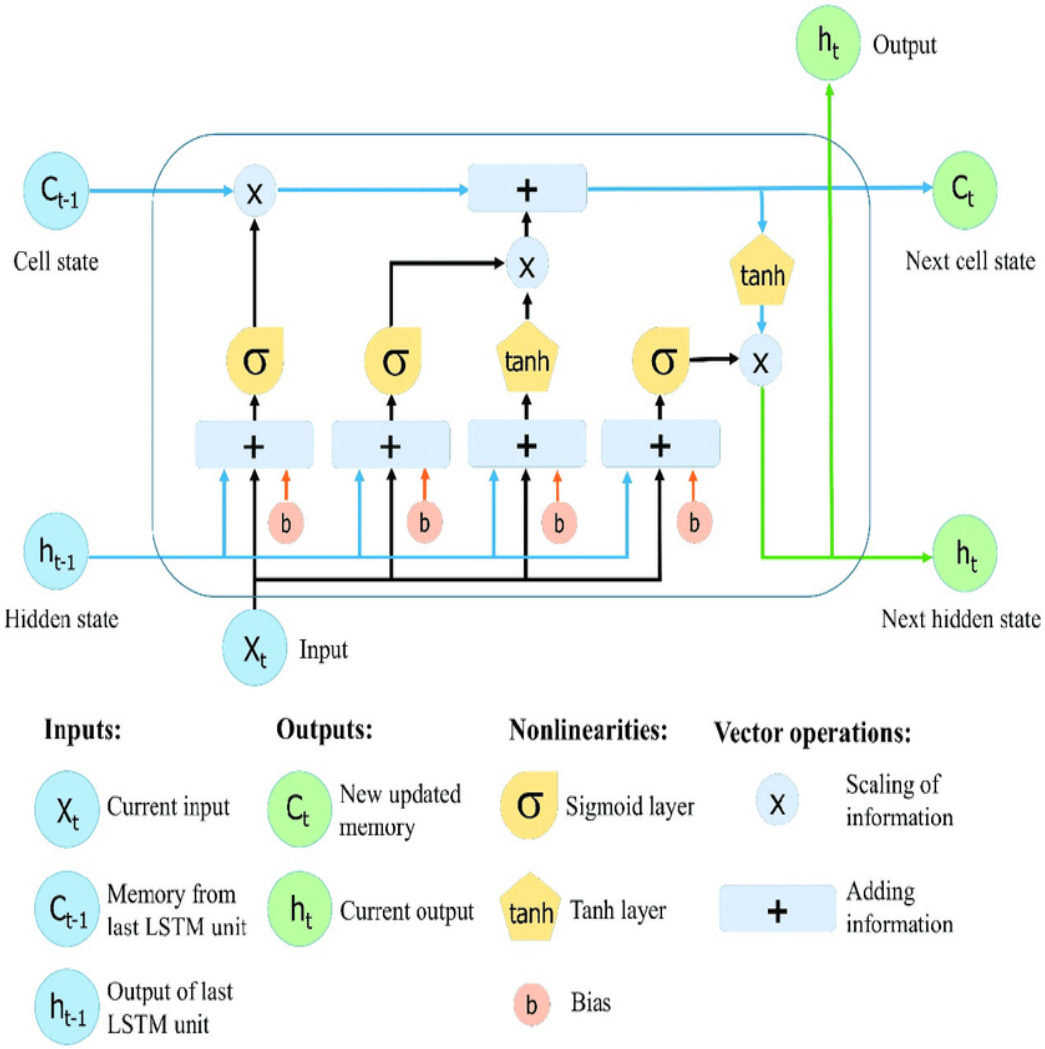

Figura 2.4: Esquema de funcionamiento de una célula LSTM.

Recuperado de *https://www.researchgate.net/figure/The-structure-of-the-Long-Short-Term-Memory-LSTM-neural-network-Reproduced-from-Yan_fig8_334268507*

3: Técnicas y herramientas

En este capítulo haremos una descripción más profunda y detallada acerca de las técnicas que se han utilizado para el desarrollo de este trabajo, además de las herramientas que han permitido su realización.

3.1. Técnicas y herramientas de desarrollo

En primer lugar debemos destacar la utilización de las dos técnicas fundamentales pertenecientes al campo del Deep Learning para este trabajo, las redes neuronales profundas, y las redes neuronales recurrentes. Para poder probar la validez del uso de RNN, se han utilizado para elaborar un modelo preliminar, y así estudiar el comportamiento del conjunto de datos que poseemos en este tipo de red. Para poder llevar esta tarea a cabo, se ha realizado una preparación de datos y adaptación de los mismos a los requisitos que impone este tipo de redes, ya que su uso requiere de estas tareas para que los modelos funcionen correctamente.

Por otro lado, se han utilizado las redes neuronales profundas para poder realizar una comparación entre los 2 tipos de redes neuronales probadas. Además, como podremos observar en la sección de resultados, este tipo de modelo ha demostrado comportarse de una mejor forma con los datos que disponemos, y avanzar en el desarrollo de este trabajo utilizando este tipo de redes neuronales. Además de ello, dentro de este tipo de solución se ha realizado un estudio de los mejores valores para los hiperparámetros de la red, con el objetivo de aumentar la precisión en los modelos creados, como podremos comprobar en la sección 4.3.

Respecto a las herramientas utilizadas en este trabajo, podemos destacar 2 grupos: las herramientas de desarrollo, y las bibliotecas utilizadas para los modelos de Deep Learning. Las herramientas de desarrollo son las siguientes:

- **Jupyter**. Esta herramienta ha sido diseñada para desarrollar software de código abierto, estándares abiertos y servicios para computación interactiva en docenas de lenguajes de programación. El servicio principal que utilizaremos de esta herramienta serán los *Jupyter Notebooks*, un entorno informático interactivo que permite crear documentos en forma de cuadernos, que incluyen diferentes funcionalidades como código en vivo, gráficos, comentarios, cabeceras, documentos de vídeo e imágenes, etc.
- **PySpark**. PySpark es una interfaz que pertenece a Apache Spark en el lenguaeje de programación Python. De entre todas las características que incluye, la principal que utilizaremos en este trabajo será *Spark SQL and Dataframe*, un módulo especializado para el procesamiento de datos estructurados, que proporciona una abstracción de programación llamada DataFrame, mediante el cual podremos estructurar los datos de una manera más sencilla y tratarlos correctamente para el uso de las técnicas de Deep Learning utilizadas, además de que también puede actuar como motor de consulta SQL distribuido para la consulta y transformación de datos.
- **Python**. Será el lenguaje de programación utilizado, ya que se complementa muy bien con las herramientas descritas anteriormente, además de ser un lenguaje interpretado, dinámico y multiplataforma.

La elección principal de estas herramientas de desarrollo viene dada por la sencillez, la alta comprensión de los elementos, la flexibilidad, y las facilidades que aportan estas herramientas, entre otras razones. Principalmente, la estructura de *Jupyter* nos permite separar los elementos en módulos independientes, sin necesidad de ejecutar todo el código desarrollado cada vez que se añade una funcionalidad, por lo que le convierte en una herramienta esencial y muy útil para realizar trabajos de este tipo. Por otro lado, *Pyspark* nos ofrece una gran solvencia y flexibilidad a la hora de operar con diferentes tipos de datos, además de contar con herramientas sencillas para tratar dichos datos, y realizar análisis y transformaciones que resultarán relevantes en nuestro estudio. Por último, como ya se ha comentado, *Python* es el lenguaje de programación más popular utilizado para el estudio de la ciencia de datos, además de ser un lenguaje sencillo y fácil de comprender, lo que nos permitirá realizar este trabajo sin necesidad de dedicar una gran cantidad de tiempo al funcionamiento de este lenguaje y a la comprensión de su estructura.

3.2. Bibliotecas de Deep Learning utilizadas

Por otro lado, debemos de destacar las bibliotecas de Deep Learning utilizadas en el desarrollo de este trabajo, además del por qué de su utilización, ya que existen multitud de librerías que pueden usarse en este tipo de problemas:

- **Tensorflow**. Herramienta de código abierto que ha sido diseñada para realizar tareas de aprendizaje automático, y que permite satisfacer las necesidades de los sistemas actuales capaces capaces de construir y entrenar redes neuronales para detectar y descifrar patrones y correlaciones.
- **Keras**. Esta herramienta es una biblioteca de Redes Neuronales de código abierto escrita en Python. Ha sido especialmente diseñada para posibilitar la experimentación en más o menos poco tiempo con redes de Aprendizaje Profundo sobre sistemas sin una gran potencia de cálculo. La elección de esta biblioteca viene dada debido a la multitud de ventajas que ofrece en tareas relacionadas con las redes neuronales, admás de ser una herramienta sencilla para el usuario, modular, y extensible entre diferentes plataformas.
- **Numpy**. Otra herramienta fundamental para el desarrollo de trabajos sobre Deep Learning, debido a que permite la creación de vectores y matrices grandes multidimensionales, usados para satisfacer los requisitos de las entradas de las redes neuronales, además de contar con una gran colección de funciones matemáticas de alto nivel para operar con ellas.
- **Pandas**. Esta herramienta es una extensión de la librería *Numpy*, creada fundamentalmente para la manipulación y análisis de datos, ya que entre sus principales funciones, ofrece estructuras de datos y operaciones para manipular tablas numéricas y series temporales de una manera sencilla, lo que hace que sea una herramienta muy útil a la hora de tratar grandes cantidades de información.
- **ScikitLearn**. Diseñada para operar con otro tipo de herramientas numéricas como *Numpy*, nos permitirá evaluar el rendimiento de los modelos creados aplicando diferentes tipos de métricas sobre ellos, ya que incluye múltiples funciones orientadas para este tipo de tareas.

La elección de todos estos tipos de herramientas se debe a las necesidades del desarrollo del problema planteado, ya que algunas de ellas se consideran esenciales para su ejecución. Por otro lado, respecto a las bibliotecas utilizadas, son las más populares en el ámbito del Machine Learning y Deep Learning, gracias a la gran capacidad que ofrecen para la resolución de problemas de diferentes tipos dentro de estos ámbitos [21]

3.3. Herramientas alternativas

Actualmente existen un sin fín de herramientas y librerías que nos permitirían realizar soluciones muy completas a problemas relacionados con el Deep Learning. Es por ello que, aun que no hayan sido utilizadas en este trabajo, debemos de poner el foco en algunas de ellas, para poder enfocar el problema desde un punto diferente si fuera necesario.

- **PyTorch**. Es un entorno de trabajo específico para el Deep Learning de código abierto desarrollado por Facebook y escrito en Python. Se basa en la biblioteca Torch y se diseñó con un objetivo principal: agilizar todo el proceso desde la creación de prototipos de investigación hasta el despliegue de producción. Esta herramienta funciona con un gráfico que se actualiza dinámicamente, lo que significa que puede realizar los cambios necesarios en la arquitectura del modelo durante el proceso de entrenamiento. Una de sus ventajas principales es que es excelente para la formación, la construcción y el despliegue de pequeños proyectos y prototipos, además de estar ampliamente utilizado para aplicaciones de Deep Learning como el procesamiento del lenguaje natural y la visión por ordenador.

- **Theano**. Es una herramienta desarrollada en la Universidad de Montreal, escrita en Python y centrada en NVIDIA CUDA, lo que permite a los usuarios integrarlo con el GPS. La biblioteca de Python permite a los usuarios definir, optimizar y evaluar expresiones matemáticas que implican matrices multidimensionales. Es una alternativa perfecta para tratar de resolver problemas matemáticos complejos que necesiten un coste computacional elevado para su desarrollo.

- **Swift for TensorFlow**. Es una plataforma de nueva generación que combina la potencia de TensorFlow con la del lenguaje de programación Swift. Al estar diseñado específicamente para el Deep Learning, Swift for TensorFlow incorpora todas las últimas investigaciones en ML, programación diferenciable, compiladores, diseño de sistemas y mucho más. Incluye una gran variedad de herramientas para ayudar a mejorar la productividad de los usuarios. Puede ejecutar Swift de forma interactiva en un cuaderno Jupyter y obtener sugerencias de autocompletado para seguir explorando la superficie de la API. Además, es una buena opción si los lenguajes dinámicos no son adecuados para sus proyectos. Al ser un lenguaje de tipado estático, Swift muestra cualquier error en el código por adelantado, para que pueda ser corregido antes de ejecutar el código.

Por otro lado, debemos destacar también los diferentes entornos de trabajo que pueden sernos de ayuda y cuáles son las principales características que ofrecen.

- **Google Collaboratory**. Esta herramienta es un producto de Google Research. Permite a cualquier usuario escribir y ejecutar código de Python en el navegador. Es especialmente adecuado para tareas de aprendizaje automático, análisis de datos y educación. Desde un punto de vista más técnico, Colab es un servicio alojado de Jupyter Notebook que no requiere configuración y que ofrece acceso gratuito a recursos informáticos, como las GPUs. Además, su uso es gratuito, lo que implica que los recursos no estén garantizados ni sean ilimitados, dónde los límites de uso a veces puedan variar. Estas restricciones son necesarias para que Colaboratory pueda ofrecer recursos de forma gratuita, pero también son la principal desventaja, ya que si trabajamos con volúmenes grandes de datos y necesitamos una capacidad de procesamiento más alta de lo normal, es posible que encontremos ciertos problemas a la hora de realizar tareas como el entrenamiento, la carga de ficheros, y las transformaciones de los datos.

- **Entorno Local**. Existen diferentes formas de trabajar con herramientas de Deep Learning, y el entorno local es una de ellas. Se pueden instalar en el ordenador personal todas las librerías necesarias y las herramientas adecuadas para trabajar, configurando así un espacio personal que no depende de recursos externos, si no del propio hardware alojado en la máquina local. La mayor desventaja que presenta esta forma de trabajo es la instalación correcta y la puesta en funcionamiento de todas las herramientas, módulos y bibliotecas necesarios para trabajar, ya que no hay un servicio externo que pueda ofrecerlo, como Google Collaboratory, pero una vez realizado, es de las opciones con mayores ventajas, ya que cuentas con el especio para trabajar siempre que lo necesites, y con la capacidad de procesamiento más que suficiente con las carácterísticas que ofrecen los ordenadores hoy en día.

- **Docker**. Docker es una plataforma de software para crear aplicaciones basadas en contenedores, es decir, entornos de ejecución pequeños y ligeros que hacen un uso compartido del núcleo del sistema operativo pero que, por lo demás, se ejecutan de forma aislada. Para ello, es necesario instalar Docker en la máquina local y, una vez realizado, instalar todas las herramientas necesarias para llevar a cabo el trabajo. Dependiendo de su sistema operativo, los procedimientos pueden variar. El uso de Docker es la forma más sencilla de configurar la compatibilidad con la GPU, y ofrecer mayores niveles de computación si fuera necesario, ya que en el entorno local no puedes llegar a configurar ese tipo de aspectos.

4: Desarrollo del trabajo de investigación

En este capítulo se describirán los aspectos más relevantes e interesantes del desarrollo del trabajo de investigación, exponiendo las partes más importantes del mismo y explicando detalladamente los caminos tomados para las soluciones y la justificación de los mismos. Además de las explicaciones, se incluirán documentos gráficos y estadísticos para realizar una completa explicación del contenido. Todos los documentos asociados al desarrollo del proyecto podrán encontrarse en el siguiente enlace: https://github.com/aitorojeda/DatosTFM.git ; debidamente estructurado en carpetas, entre ficheros de datos y código de desarrollo.

4.1. Preparación del entorno de ejecución

Antes de comenzar con la preparación de los datos, debíamos de configurar un entorno de trabajo estable para poder realizar todas las operaciones necesarias sobre los datos, además de realizar todas las tareas del trabajo de investigación sobre dicho entorno.

El entorno de trabajo que se utilizará para el desarrollo será Anaconda Navigator, un entorno local que nos permitirá desplegar varias herramientas de trabajo diseñadas específicamente para la ciencia de datos, como Jupyter, Pycharm y Spyder entre otras muchas. Para el desarrollo del código, utilizaremos cuadernos de Jupyter, usando como módulo principal Pyspark, y Python como lenguaje de programación. Las tareas principales que se han llevado a cabo para la puesta a punto de este entorno de trabajo son las siguientes:

- Descarga del entorno de trabajo Anaconda Navigator
- Creación de un entorno propio de ejecución local
- Instalación de las librerías principales con las que se van a trabajar, explicadas previamente en el Capítulo 2.2.
- Ejecución y comprobación del funcionamiento de los diferentes modulos a utilizar

En el Apéndice B se explicará con más detalle los pasos a seguir para la instalación, además de los comandos necesarios para poner a punto el entorno de trabajo y comenzar a trabajar con él.

4.2. Preparación de datos. Exploración, limpieza y transformación

En primer lugar, para empezar a trabajar en un modelo de red neuronal, debemos de adaptar nuestros datos, explorar los diferentes valores que pueden tomar los mismos, limpiarlos de registros corruptos, inconsistentes o vacíos, y transformarlos, para así poder introducirlos dentro de una red neuronal de la manera más correcta y completa posible, ya que éstos modelos requieren ciertos tipos de datos, y estructurados de diferentes formas para comenzar a realizar el aprendizaje.

Exploración de los datos

Nuestro conjunto de datos original consta de una serie de vectores formados a partir de velocidades angulares y de aceleraciones, convenientemente etiquetados con tiempo en microsegundos. Un total de 47 personas (13 mujeres y 34 hombres) de entre 18 y 68 años colaboraron en la elaboración de este conjunto de datos. Independientemente de si el sujeto era diestro o zurdo, se pidió a los individuos que interactuaran de la manera más natural posible con una manija de puerta para diestros durante 20 repeticiones en una puerta firmemente cerrada.

La plataforma de adquisición de datos contaba con un pequeño módulo MEMS conectado a la manija de la puerta (de palanca común) controlada por una pequeña computadora fijada a la puerta. La plataforma está formada por una Raspberry Pi como placa principal 2 +, junto con el elemento principal que actúa como sensor, una MPU 9250 (Unidad de procesamiento de movimiento) que se puede encontrar en algunos teléfonos inteligentes. En la Figura 4.5 podemos observar cómo se colocaron dichos dispositivos en la puerta, adhiriendo a la manilla de la misma mediante un cableado especial conectado a la raspberry.

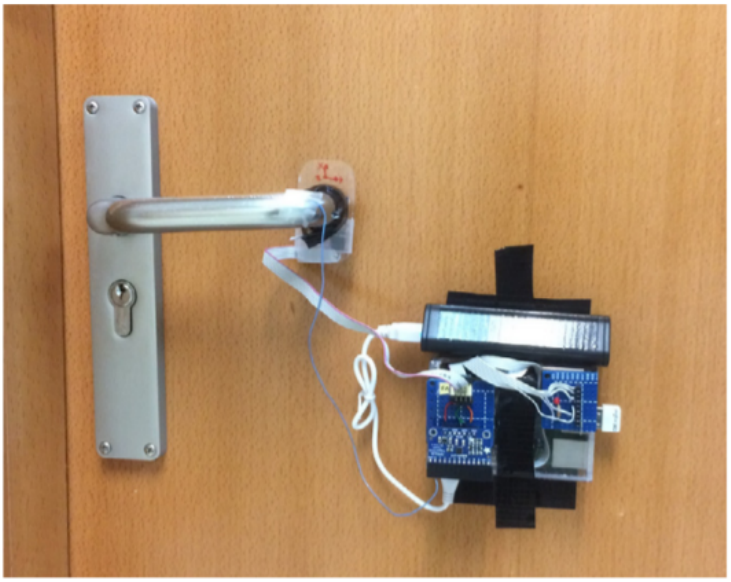

Figura 4.5: Despliegue de lataforma de adquisición.

Se pidió a los sujetos que intentaran abrir una puerta cerrada, y para ello, debían de seguir los siguientes pasos:

1. Comenzar a una distancia de 3 m justo en frente de la puerta.
2. Caminar como lo hacen habitualmente con la intención de actuar sobre la manija de la puerta.
3. Actuar sobre la manija de la puerta
4. Esperar un tiempo muy corto con la palanca en la posición de tope final
5. Liberar la fuerza mientras se maneja la manija de la puerta
6. Alejar la mano

En relación a estas etapas, debemos de destacar el comportamiento de los usuarios con los datos. En la figura 4.6, en las gráficas correspondientes a la aceleración en el eje X (a(g)), y a la velocidad angular en el eje Z (w(0/s)), observamos las diferentes etapas que compone el proceso de toma de datos con 3 usuarios distintos, y como varían los datos a medida que los usuarios interaccionan con la manilla de la puerta.

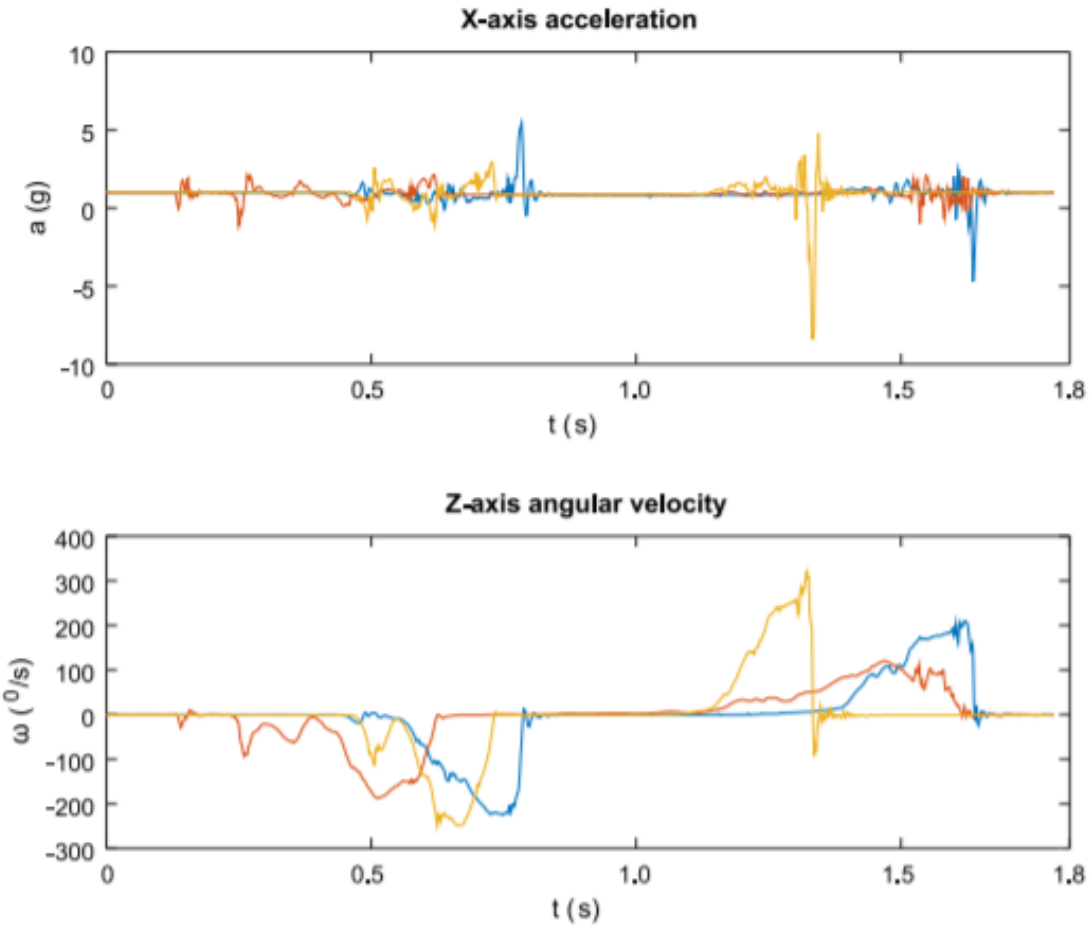

Figura 4.6: Aceleración (Ax(g)) y velocidad angular (Wz(0/s)) de los datos adquiridos de tres sujetos de estudio diferentes, que interactúan con la manija de la puerta siguiendo las instrucciones dadas

.

Una vez definida la fuente de datos de nuestro trabajo, debemos indicar la estructura en la que se encuentran los mismos. Los datos del corpus están formados por 47 ficheros, divididos por cada usuario, del 1 al 47, y dentro de cada directorio de los usuarios, se encuentran 20 ficheros, uno por cada intento del usuario que se ha descrito anteriormente. En total contamos con 940 ficheros de datos, en formato csv, con 1495 registros por cada fichero. Las columnas que conforman cada fichero son las siguientes:

Nombre	Tipo de valor	Tipología	Descripción
num	Int	Cuantitativo discreto	Número de registro
time(us)	Int	Cuantitativo continuo	Tiempo en microsegundos
accX(g)	Float	Cuantitativo continuo	Aceleración en eje X
accY(g)	Float	Cuantitativo continuo	Aceleración en eje Y
accZ(g)	Float	Cuantitativo continuo	Aceleración en eje Z
gyrX(o/s)	Float	Cuantitativo continuo	Velocidad angular en X
gyrY(o/s)	Float	Cuantitativo continuo	Velocidad angular en Y
gyrZ(o/s)	Float	Cuantitativo continuo	Velocidad angular en Z

Tabla 4.1: Descripción de los atributos

Debido a la complicación que supone tener separados los ficheros de datos, para una mayor comodidad y mejor análisis, se ha procedido a juntar todos y cada uno de los ficheros que forman el corpus de datos en un único fichero con el que trabajaremos posteriormente. Con todos los datos concatenados en un fichero, podemos observar en la figura 4.7, una descripción estadística de los mismos de carácter general, con el número de registros totales, la media, la desviación estándar, el mínimo y el máximo de cada atributo, y los percentiles 0.25, 0.50, y 0.75. Si los datos los hubiéramos tratado de manera independiente, este análisis no habría resultado relevante en el estudio ya que los valores obtenidos serían para cada fichero de datos, y no de una manera más general.

	accX(g)	accY(g)	accZ(g)	gyrX(o/s)	gyry(o/s)	gyrz(o/s)
count	1.405300e+06	1.405300e+06	1.405300e+06	1.405300e+06	1.405300e+06	1.405300e+06
mean	9.675167e-01	-1.702909e-01	1.718306e-02	8.716722e-01	4.921452e-02	-7.559592e-01
std	5.337054e-01	2.643577e-01	2.076064e-01	4.270605e+00	1.050591e+01	6.357781e+01
min	-1.593848e+01	-1.027734e+01	-1.501758e+01	-3.602290e+02	-4.930992e+02	-4.528092e+02
25%	8.828120e-01	-4.663090e-01	-3.418000e-03	5.496180e-01	-2.290080e-01	-1.175573e+00
50%	1.005859e+00	-2.246100e-02	1.611300e-02	8.702290e-01	4.580200e-02	-7.633590e-01
75%	1.014648e+00	6.348000e-03	3.564500e-02	1.114504e+00	2.900760e-01	-5.343510e-01
max	1.599951e+01	8.896484e+00	1.548096e+01	1.594504e+02	4.930534e+02	4.970076e+02

Figura 4.7: Descripción estadística de los datos de las coordenadas

Limpieza y transformación de los datos

Para continuar con la preparación de los datos, se ha realizado un análisis para cada atributo en el que se comprueban registros vacíos o incorrectos. Gracias a dicho análisis, se ha podido observar que todos los datos son correctos, y no incluyen ningún registro vacío, debido a que previamente, en el trabajo anteriormente mencionado, se han ocupado del tratamiento completo de los datos.

El último punto de preparación de los datos es la transformación que se ha realizado sobre los mismos. Los ficheros de datos incluían en el nombre un número, que indicaba el intento del usuario asociado al mismo. También, dichos ficheros estaban recogidos en directorios donde se indicaba el número de cada usuario. Es por ello que, para seguir conservando esta información, se han incluído 2 atributos más al conjunto de datos, para cada uno de los ficheros del corpus:

Nombre	**Tipo de valor**	**Tipología**	**Descripción**
User_ID	Int	Cuantitativo discreto	Número que identifica al usuario
Attemp	Int	Cuantitativo discreto	Intento de cada usuario

Tabla 4.2: Descripción de los nuevos atributos añadidos

Estos atributos nos servirán como clase para poder identificar a los usuarios, y como claves de partición para reducir el conjunto de datos y probar diferentes combinaciones de intentos. El rango que pueden alcanzar estos dos atributos son:

- User_ID. De 1 a 47.
- Attemp. De 1 a 20.

Además de estas transformaciones, se ha procedido a borrar del conjunto de datos la columna denominada “num", debido a que no aportaba ningún tipo de relevancia al conjunto de datos y simplemente era un identificador de registros, con un rango de 1 a 1495 en cada fichero de datos.

4.3. Protocolo experimental

En primer lugar vamos a realizar la descripción de los diferentes modelos de Deep Learning que se han creado durante la etapa de investigación, con el objetivo claro de probar nuestra hipótesis principal: utilizar redes neuronales para realizar una autenticación de usuarios. Durante el comienzo de la fase de desarrollo, se plantearon diversas soluciones iniciales para el problema estudiado, siendo la Tabla 4.3 una pequeña representación de las diferentes fases que se han abordado durante el desarrollo.

Fase del desarrollo	**Descripción**
Modelo inicial	Creación del primer modelo utilizado
Investigación de parámetros	Estudio y optimización de parámetros
Evaluación del modelo	Evaluación del modelo con diferentes técnicas

Tabla 4.3: Fases del protocolo experimental

Modelo inicial

En esta etapa se han llevado a cabo diferentes operaciones con el objetivo de la creación de un primer modelo que nos pudiera servir de referencia para nuestro objetivo principal. Para ello, se ha elaborado un modelo con una capa "LSTM" (Long Sort-Term Memory). Como ya se ha explicado en el capítulo 1.4, la arquitectura principal de este modelo se basa principalmente en redes neuronales recurrentes, lo que nos permitirá realizar un modelo adaptado a series temporales con los datos que poseemos.

Se ha utilizado este tipo de capa sobre la red neuronal, para comprobar cómo era su funcionamiento dentro de la red y estudiar cómo se comportaría la red aplicando este tipo de técnica, debido a que no se había realizado ningún estudio previo de los parámetros a aplicar. El resultado obtenido no fue demasiado bueno, debido a que se obtenía más o menos un 38 % de precisión, lo que no es un resultado relevante para la investigación realizada. Por ello, se decidió dejar apartado este tipo de técnica y hacer un estudio más profundo acerca de las necesidades, parámetros y valores que necesitaba el modelo.

Estudio y optimización del modelo

Conocido el funcionamiento preliminar del modelo, realizamos un pequeño estudio acerca de los métodos y parámetros que mejor se ajustarían al problema estudiado. Nuestra hipótesis principal es investigar acerca de si es posible mediante técnicas de deep learning realizar una autenticación de usuarios. Por tanto, el problema que mejor se ajusta a esta hipótesis es el enfoque multiclase, en el cual tenemos diferentes datos para cada una de las clases, y queremos hacer una clasificación más o menos precisa si la red es capaz de identificar a los usuarios con los datos suministrados para ello. Siguiendo esta línea, las clases hacen referencia a los diferentes usuarios que tenemos, y los atributos estudiados (las columnas de los ficheros de datos) son las características a tener en cuenta de cada usuario, es decir, los datos que alimentarán el modelo para realizar dicha clasificación. Los parámetros utilizados son los siguientes:

- **Tipo de modelo**. Un modelo es la estructura de datos básica de Keras, la librería de deep learning utilizada para este trabajo. Los modelos de Keras definen cómo organizar las capas del modelo,en este caso, creándolas en orden secuencial, lo que nos permite crear modelos capa por capa en dicho orden. Pero uno de los inconvenientes es que no nos permite crear modelos que tengan múltiples entradas o salidas, siendo

buena esta aproximación para una pila simple de capas que tienen 1 tensor de entrada y 1 tensor de salida, como es este caso.

- **Función de activación.** La función de activación de un nodo define la salida de ese nodo dada una entrada o conjunto de entradas. Un circuito integrado estándar puede verse como una red digital de funciones de activación que pueden estar en"ON" (1) u "OFF" (0), dependiendo de la entrada. Esto es similar al perceptrón lineal de las redes neuronales. Sin embargo, sólo las funciones de activación no lineales permiten a estas redes calcular problemas no triviales utilizando sólo un pequeño número de nodos, y tales funciones de activación se denominan no lineales.

- **Optimizador**.Los optimizadores son algoritmos o métodos utilizados para minimizar una función de error (función de pérdida) o para maximizar la eficiencia de la producción. Consisten en funciones matemáticas que dependen de los parámetros de aprendizaje del modelo, es decir, los pesos y los sesgos. Los optimizadores ayudan a saber cómo cambiar los pesos y la tasa de aprendizaje de la red neuronal para reducir las pérdidas. Existen múltiples algoritmos de optimización: Adam, rmsprop, adagrad, SGD, nadam, etc.

- **Función de pérdida**. Hace referencia al parámetro que nos permite minimizar el error de la función de optimización. La función de pérdida tiene un trabajo importante, ya que debe destilar fielmente todos los aspectos del modelo en un solo número, de tal manera que las mejoras en ese número sean un signo de un mejor modelo. También existen múltiples funciones de pérdida, cada una adaptada al problema que se plantea: Regression Loss Function, Mean Squared Error, Mean Squared Logarithmic Error Loss, Mean Absolute Error Loss, Binary Classification Loss Function, Binary Cross Entropy Loss, etc. En este caso, utilizaremos la función *Categorical cross-entrophy*, debido a que es la función de pérdida que más se ajusta al problema que estamos tratando.

- **Épocas.** El número de épocas es un hiperparámetro que define el número de veces que el algoritmo de aprendizaje trabajará a través de todo el conjunto de datos de entrenamiento. Una época significa que cada muestra del conjunto de datos de entrenamiento ha tenido la oportunidad de actualizar los parámetros del modelo interno. Este número es variable y depende del problema y el volumen de datos, se puede establecer un valor mayor o menor.

- **Tamaño del lote**. El tamaño del lote es un hiperparámetro que define el número de muestras con las que se trabaja antes de actualizar los parámetros internos del modelo. Es por ello que un lote es como un bucle "for" que itera sobre una o más muestras y hace predicciones. Al final del lote, las predicciones se comparan con las variables de salida esperadas y se calcula un error. A partir de este error, el algoritmo de actualización se utiliza para mejorar el modelo, por ejemplo, para moverse hacia abajo a lo largo del gradiente de error. Este hiperparámetro puede ser de varios tipos:

- Descenso de gradiente por lotes. Tamaño del lote = Tamaño del conjunto de entrenamiento
- Descenso de gradiente estocástico. Tamaño del lote = 1
- Descenso de gradiente en mini lotes. 1 <Tamaño del lote <Tamaño del conjunto de entrenamiento

Con los hiperpárametros ya descritos que se van a utilizar para la creación de un modelo más ajustado a nuestro problema, podemos realizar algunas observaciones acerca de los valores más óptimos de algunos de ellos. Respecto a la función de activación, los valores elegidos son las funciones "Softmax" y "Relu", debido a que la primera de ellas asegura que los valores de salida se encuentran en un rango del 0 al 1, para poder realizar la predicción de las clases correctamente, y la segunda, debido a que es la función más conocida gracias a su simplicidad en el cálculo de los pesos. Además el probblema tratado es de tipo multiclase, y son las funciones que mejor se ajustan para dicho problema.

Sin embargo para el optimizador, se han utilizado diferentes valores, pero el que mejor se comportaba es el optimizador `adam`, debido a que la función de pérdida y la función de precisión, tienen un mayor relación y la distribución es más representativa, ya que por ejemplo con el optimizador `rmsprop`, la función de pérdida alcanzaba picos en el entrenamiento del modelo, y los valores no eran demasiado buenos, como podemos observar en la siguientes gráficas.

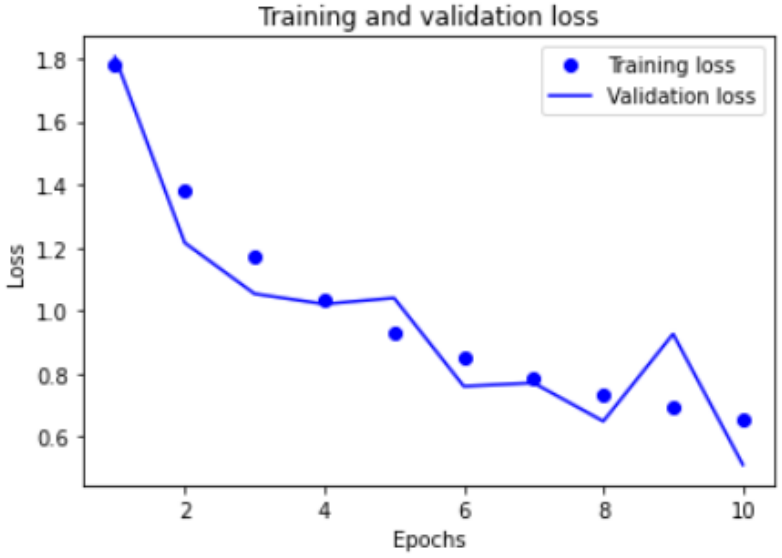

Figura 4.8: Función de pérdida con el optimizador Rmsprop

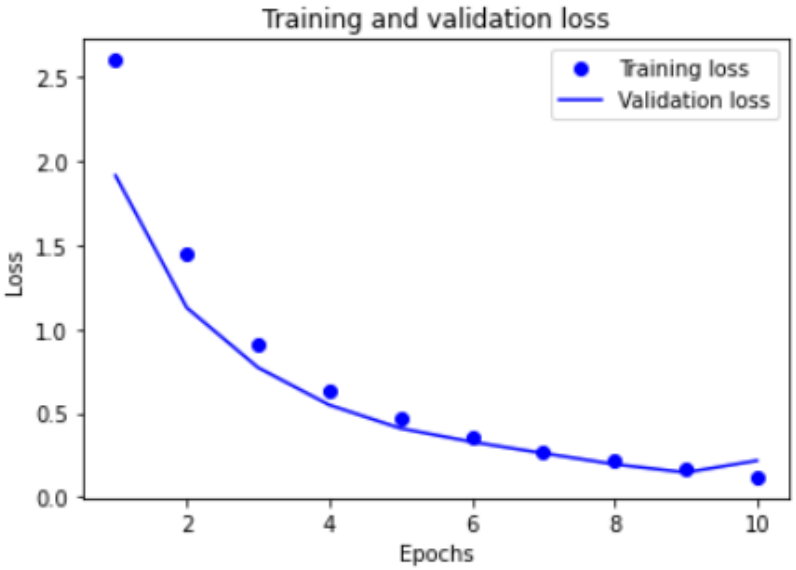

Figura 4.9: Función de pérdida con el optimizador Adam

Por otro lado, también se han probado una combinación de diferentes valores en el hiperparámetro "épocas". Debido a que tenemos una gran cantidad de datos, y con un grado alto de similitud entre ellos, no podemos utilizar un número elevado de las mismas debido a que podríamos encontrarnos con el problema del sobreajuste o sobreentrenamiento. Esto quiere decir cuando se modela los datos de entrenamiento demasiado bien, aprendiendo detalles de estos que no son generales. Esto es debido a que sobreentrenamos nuestro modelo y éste estará considerando como válidos solo los datos idénticos a los de nuestro conjunto de entrenamiento, incluidos sus defectos. Por tanto, los mejores valores para este hiperparámetro son entre 5 y 12.

También, debemos destacar los mejores valores estudiados para el tamaño del lote. Éstos deben de ser valores medianamente elevados, debido a que el volumen de datos disponibles son muchos, y si el tamaño del lote es pequeño, el entrenamiento de cada modelo puede tardar demasiado. Es por ello que los valores aproximados que se utilizarán para evaluar el modelo serán 512 y 1024, dependiendo del volumen de datos que tengan los conjuntos de entrenamiento y test.

4.4. Pruebas y evaluación de los modelos

Para concluir con la parte experimental del proyecto, en esta sección se recogerán las distintas evaluaciones realizadas sobre los modelos de redes neuronalesque se han creado. Utilizando algunos de los mejores valores de los hiperparámetros estudiados en la sección anterior, se ha elaborado una tabla de evaluación de modelos en la que se ha utilizado como herramienta de evaluación el método de *StratifiedKFold*. Este método está basado en la herramienta conocida como *Cross validation* o *KFold*, que consiste en evaluar un modelo con una serie de particiones en los datos de entrenamiento y prueba. Sin embargo, StratifiedKFold es una variación que devuelve "pliegues" estratificados. Los pliegues se realizan conservando el porcentaje de muestras para cada clase, y proporciona índices de entrenamiento/prueba para dividir los datos en conjuntos de entrenamiento/prueba. Este método está más orientado a los problemas binarios o multiclase, que es justamente el problema que se está tratando en este trabajo.

Para la creación y evaluación de los modelos, se han tenido en cuenta los valores de los hiperparámetros estudiados en la sección anterior, además de 2 puntos fundamentales respecto a los datos:

- En primer lugar, se han escogido 5, 10, y 20 intentos de cada usuario para comprobar cómo se comporta el modelo con diferentes volúmenes de datos y encontrar un tamaño de datos adecuado para el problema estudiado.
- En segundo lugar, se han probado diferentes combinaciones de las columnas que componen cada fichero de datos, en concreto, 3 combinaciones, las 6 coordenadas asociadas a la velocidad angular y a la aceleración, la velocidad angular en el eje Z y la aceleración en el eje X; y sólo la velocidad angular en el eje Z. Tanto la velocidad angular en el eje Z, como la aceleración en el eje X, se han escogido debido a que, como se estudia en el artículo [7], son las dos coordenadas que más información aportaban de cada usuario, debido a que las demás no sufrían una variación importante entre los diferentes usuarios estudiados. Las columnas pertenecientes al identificador del usuario y al identificador del intento por cada usuario se han mantenido en todas las pruebas, solo se han variado la combinación de los datos asociados a las coordenadas.

Métricas

Las métricas utilizadas para la evaluación del rendimiento han sido varias, medidas en base los índices de la matriz de confusión, para poder comprobar desde diferentes aspectos cómo se comportan los modelos y sacar conclusiones más apropiadas. En la Figura 4.10 se muestran los cuatro índices cuando se comparan las etiquetas reales y las etiquetas predichas en la clasificación: positivo verdadero (TP), negativo verdadero (TN), falso positivo (FP) y falso negativo (FN).

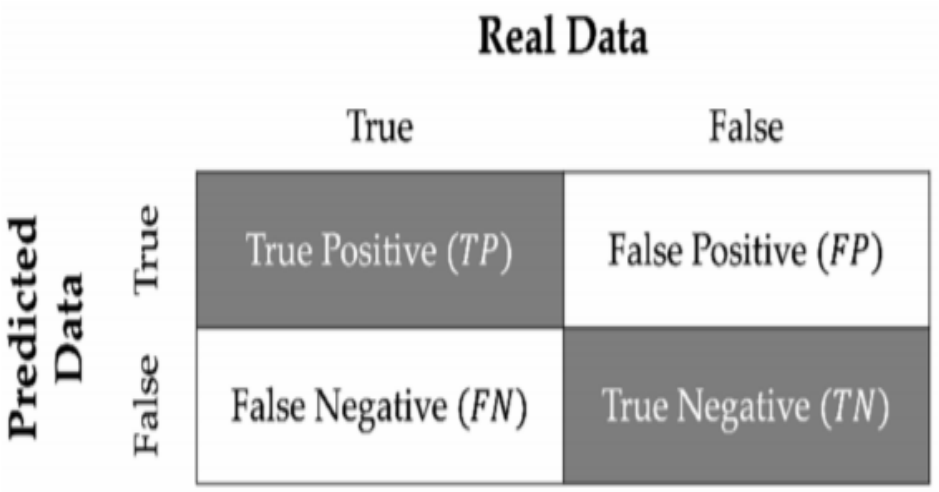

Figura 4.10: Matriz de confusión

[9].

Por otro lado, la Tabla 4.4 muestra las métricas de evaluación del modelo utilizadas, además de sus ecuaciones para los modelos de utilizados en el presente estudio, incluyendo *Accuracy*, *Precision*, *Recall* y *F1 Score*.

Métrica	**Ecuación**
Accuracy	$(TP + TN) \div (TP + FN + FP + TN)$
Precision	$(TP) \div (TP + FP)$
F1 Score	$2 \times (Recall \times Precision) \div (Recall + Precision)$
Recall	$(TP) \div (TP + FN)$

Tabla 4.4: Métricas de evaluación

Resultados

A continuación, se presentarán los resultados obtenidos de la evaluación de los diferentes modelos mediante las métricas explicadas anteriormente, separando los modelos creados en 3 tipos: 5 , 10 y 20 intentos por cada usuario. Las siguientes tablas mostradas a continuación muestran los resultados obtenidos de la creación de distintos modelos realizando diferentes combinaciones con el tamaño del lote, además de comprobar su funcionamiento con las diferentes combinaciones de columnas que contienen los datos del corpus.

Intentos	**Nº Coordenadas**	**Accuracy**	**Precision**	**Recall**	**F1 Score**
5	1	80.83	85.06	80.83	77.97
	2	**81.89**	**85.26**	**81.89**	**79.21**
	6	79.12	81.71	79.12	77.19
10	1	95.79	97.18	95.79	95.26
	2	**97.27**	**97.85**	**97.27**	**97.01**
	6	94.45	95.81	94.45	94.08
20	1	**99.99**	**99.99**	**99.99**	**99.99**
	2	99.91	99.91	99.91	99.91
	6	99.70	99.70	99.70	99.70

Tabla 4.5: Resultados de la clasificación con diferentes conjuntos de datos (intentos) por usuario

Como podemos observar en la Tabla 4.5, se presentan los datos para las métricas *accuracy, precision, recall* y *f1 score*, asociadas al número de intentos elegidos para el conjunto de datos (5, 10 y 20), y para el número de columnas utilizadas para la creación del modelo (1, 2 y 6), explicado anteriormente en esta sección. Los mejores valores están asociados al uso de 2 columnas en la creación de los modelos, para todos los intentos excepto el modelo de 20 intentos, que los mejores valores están asociados al uso de 1 columna. Sin embargo, los peores valores los podemos encontrar cuando usamos 6 columnas en la creación de los modelos para todos los casos, es decir, para cualquier tamaño del conjunto de datos.

Discusión

Respecto a los resultados obtenidos acerca de las ejecuciones con diferentes volúmenes de datos, se han extraído diferentes conclusiones de cada una de las mismas. En primer lugar, en el apartado de la teoría, se introdujo a los modelos de redes neuronales usados para realizar una clasificación de usuarios. Comparando el rendimiento de los diferentes modelos creados para determinados volúmenes de datos, podemos afirmar que los modelo construidos son bastante aceptables, aun que algunas de las métricas observadas pueden parecer demasiado altas, como es el caso de los modelos creados para un volumen de datos de 20 intentos por cada usuario, es decir, un total de 1.405.300 registros, siendo ésta una cantidad demasiado alta. Las métricas de estos modelos no son muy favorables a la hora de sacar conclusiones ya que, debido a que existe un volumen tan grande de datos, es posible que los modelos se hayan sobreentrenado, es decir, que hayan repetido una y otra vez datos muy parecidos en el entrenamiento, produciéndose así este efecto de métricas tan elevadas.

Por otro lado, haciendo hincapié en los modelos de 5 intentos por cada usuario, podemos observar que las métricas son parecidas entre los diferentes modelos, siendo la mejor el modelo creado con 2 coordenadas, debido a que a la hora del entrenamiento, aporta mayor información a la red neuronal, y por tanto las métricas son un poco más favorables. Obteniendo una precisión en torno al 80 %, podemos determinar que se tratan de modelos acepatables con una cantidad de datos adecuadas, pero se podrían utilizar diferentes técnicas de optimización, cómo la prueba de diferentes valores en los hiperparámetros, la normalización, el reescalado de los elementos, etc, para intentar mejorar su comportamiento y obtener métricas mayores.

Observando los modelos creados con 10 intentos por cada usuario, podemos afirmar que las métricas son muy favorables, y estaríamos hablando de los mejores modelos creados en este estudio. La cantidad de datos es bastante elevada, pero usando la misma combinación de hiperparámetros que en los otros 2 casos, observamos que las métricas son mucho mejores, con valores en las mismas en torno a una media de los modelos del 95 % de precisión, descartando el problema del sobreentrenamiento, debido a que la cantidad de datos no es tan grande, y los valores de los registros es más complicado que consigan repetirse, y que se vea reflejado en el entrenamiento de los modelos.

Generalmente, podemos destacar el rendimiento de las diferentes combinaciones de coordenadas, observando que no siempre a mayor número de datos de entrada, podremos conseguir métricas mejores, como se ve reflejado en todos y cada uno de los casos probados. Esto se debe a que la información relevante para poder determinar si un registro pertenece a un usuario o a otro, está recogida en las 2 coordenadas probadas, la aceleración en el eje X, y la velocidad angular en el eje Z, siendo los atributos del modelo que más información aportan en el entrenamiento, como se puede observar en todas las métricas estudiadas.

Teniendo en cuenta que el rendimiento de un algoritmo de deep learning depende directamente de la cantidad de datos de entrenamiento, la Figura 4.11 demuestra cómo el rendimiento de los modelos creados mejoró con el aumento del tamaño del conjunto de datos, es decir, con el aumento de intentos por cada usuario.

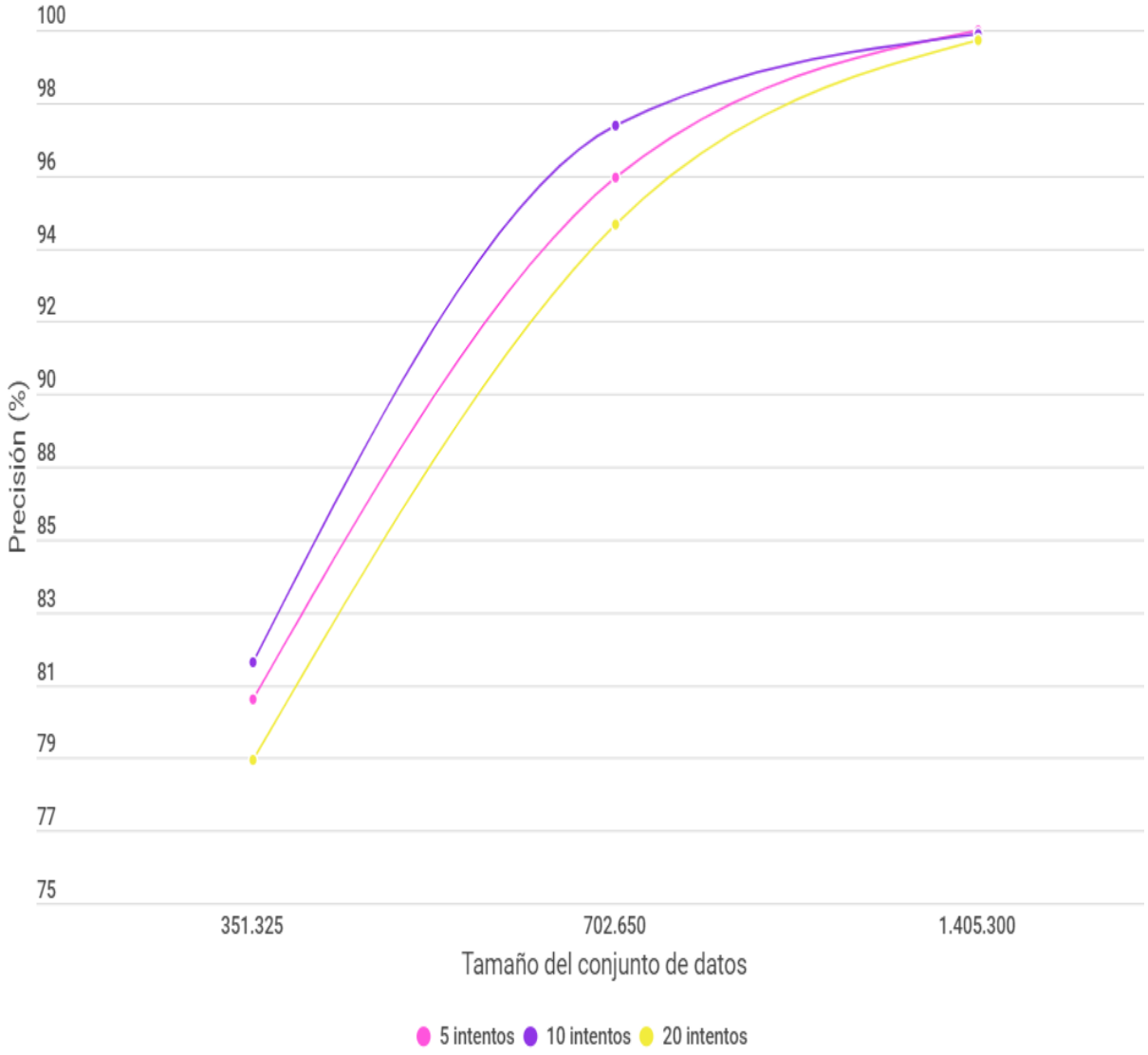

Figura 4.11: Variación de la precisión de los modelos de deep learning al aumentar el tamaño del conjunto de datos

Cada línea representa la tendencia de los modelos creados por cada intento, a medida que el tamaño del conjunto de datos aumenta, y cada punto, es un modelo diferente con la misma configuración de los parámetros. Podemos observar el gran incremento de la precisión cuando utilizamos un conjunto de datos mayor para todos y cada uno de los modelos creados, además de que, con 20 intentos, todos los modelos se comportan de una manera casi excepcional, lo que no le convierte en una buena medida para continuar con dicha configuración.

Para finalizar, debemos de tener en cuenta que los resultados obtenidos son muy favorables para el problema planteado inicialmente, la autenticación de usuarios. Destacar el buen rendimiento de los modelos, creados a partir de 351.325 y 702.650 números de registros en el conjunto de datos, con tasas de precisión de 81.89 % y 97.27 % respectivamente, que nos permite afirmar que no es necesario un conjunto de datos demasiado grande para solucionar el problema, si no que en múltiples ocasiones, un exceso de datos es perjudicial para tareas de análisis de datos de este tipo, tomando un papel más relevante la calidad de los datos que la cantidad de los mismos.

5: Conclusiones y líneas de trabajo futuras

5.1. Conclusiones

Este trabajo expone una forma concreta de realizar una autenticación de usuarios, a través de la interacción de los mismos con la manilla de una puerta, mediante la creación de modelos basados en técnicas de Deep Learning. Son varias las contribuciones de este trabajo de investigación en el ámbito descrito. En primer lugar, hasta donde podemos saber, este es el primer estudio que emplea modelos de redes neuronales profundas para realizar una autenticación de usuarios a través del mecanismo de apertura de una puerta, demostrando un rendimiento bastante bueno frente a otro tipo de técnicas de autenticación como son los algoritmos tradicionales de Machine Learning.

En segundo lugar, hemos realizado una investigación completa, acerca de las mejores técnicas, y de la mejor combinación de valores de los parámetros utilizados en las redes neuronales profundas, además de probar el funcionamiento de los modelos basados en redes neuronales recurrentes, utilizando LSTM como capa principal en los mismos. Por otro lado, también se ha realizado una preparación de datos para poder adapatarlos a los modelos de las redes neuronales utilizadas, así como una investigación sobre el origen de los mismos, y de las características que s epodían aportar para que los modelos utilizados obtuvieran un mejor rendimiento.

Además de ello, se ha propuesto un enfoque similar para el problema planteado por J. Vegas y C. LLamas [7], en su trabajo sobre la autenticación de usuarios, empleando en este estudio técnicas de Deep Learning, en vez de algoritmos más clásicos propios de Machine Learning.

En particular, se ha demostrado además cómo es el rendimiento de los diferentes modelos de redes neuronales profundas en relación con los conjuntos de datos utilizados y con el número de datos de entrada de los mismos, obteniendo un buen rendimiento para conjuntos de datos inferiores al millón de registros, con resultados de *accuracy* de 97.27 %, o *f1 Score* de 97.01 %. Por otro lado, también se ha llegado a la conclusión de que los algoritmos de Deep Learning ofrecen mejores tasas de precisión que los algoritmos de Machine Learning clásicos recogidos en el artículo [7], demostrando así su gran potencial para realizar tareas de autenticación utilizando diferentes tipos de patrones de comportamiento.

5.2. Limitaciones y trabajo futuro

Este estudio presenta algunas limitaciones que pueden considerarse como trabajo futuro. En primer lugar, destacar que el estudio se ha centrado en realizar una clasificación de usuarios, para obtener las mejores técnicas y características sobre los modelos que se deden de aplicar en una autenticación de usuarios. Esta clasificación se ha realizado en un entorno local, disponiendo de una cantidad limitada de datos. Por tanto, para poder afirmar las conclusiones descritas en este artículo, deberíamos recabar más información, y no sólo realizar un modelo en un contexto cerrado, si no que se permita realizar una autenticación con un mayor número de usuarios.

Por otro lado, otra de las limitaciones del estudio es que no se ha probado en una situación real, por tanto debemos de utilizar las técnicas y plataformas de adquisición de datos usadas en el trabajo previo a este estudio para determinar el comportamiento del modelo en un marco real de ejecución.

Además, para poder poner en práctica las técnicas utilizadas en este estudio, es necesario plantear un sistema de control de acceso en tiempo real, que nos permita realizar una autenticación de usuarios al instante, o a pocos segundos de la interacción del usuario con la manilla de la puerta, ya que las técnicas utilizadas no permiten realizarlo, debido a la gran cantidad de tiempo empleado por los modelos para evaluación de los mismos, y comprobar su funcionamiento con un conjunto pequeño de datos.

Finalmente, para poder plantear el escenario anterior, debemos de recurrir a técnicas más avanzadas de adquisición y computación de datos, utilizando computación en la nube, u entornos con gran capacidad de cálculo para que, cuando un usuario interactúe con el mecanismo, el sistema sea capaz de autenticarlo correctamente en un periodo de tiempo relativamente bajo, y que el usuario no tenga que estar esperando una respuesta por parte del sistema durante mucho tiempo. Este tipo de solución incluiría técnicas más avanzadas de procesamiento de datos, además de modelos de sistemas en tiempo real, conectado con un dispositivo de adquisición de datos más potente, es decir, la elaboración de un sitema completo para realizar la autenticación de usuarios con la información que se ha expuesto en este artículo.

Apéndices

Apéndice A

Plan de Proyecto

Para poder afrontar de manera correcta el proyecto, se ha llevado a cabo una planificación del mismo, en la que podremos distinguir diferentes fases de la planificación.

A.1. Introducción

El enfoque que vamos a tener en cuenta para el trabajo de investigación será el de "Desarrollo en cascada". Siendo uno de los modelos clásicos de desarrollo del sistema, puede ser muy favorable para el proyecto, debido a que en cada fase de la planificación, y a medida que avanza el proyecto, este modelo crea comprobaciones para verificar que se han cumplido los requisitos establecidos en el primer momento. Los jefes del proyecto al final de cada fase, además de las personas encargadas del desarrollo, son los encargados de revisar el progreso del mismo e identificar si el trabajo realizado es válido, habiéndose cumplido los requisitos establecidos. En este trabajo, podemos distinguir diferentes fases:

- Investigación. En esta actividad se han realizado labores específicas de investigación relacionadas con el Deep Learning y las Redes Neuronales, como su funcionamiento, qué tipos de datos admiten, etc.
- Preparación del entorno. En esta actividad se ha planteado un entorno de trabajo local y flexible para poder trabajar con los datos de manera cómoda y ordenada, por ello era necesario establecer un entorno de trabajo óptimo que nos permitiera la mejor manera de manipulación de los datos con los que hemos trabajado.
- Exploración de datos. En este punto se han estudiado los datos del conjunto original, los tipos con los que estaban guardados los datos,y que ltareas de preparación debían de realizarse sobre ellos.
- Adaptación de datos. Esta actividad se basa en la adaptación de los datos a las redes neuronales más conocidas, teniendo como primer ejemplo el algoritmo LSTM.

- Pruebas iniciales. Además de la adaptación de los datos y la limpieza de los mismos, también se ha realizado una pequeña prueba para comprobar la efectividad del algoritmo LSTM con los datos del corpus, sin realizar ningún tipo de modificación.
- Estudio y evaluación de parámetros. Esta tarea ha consistido en realizar una investigación profunda acerca de los mejores valores para los parámetros de las redes neuronales, además de ir comprobando su funcionamiento en un modelo de red neuronal profunda.
- Elaboración y evaluación de los modelos finales. Esta actividad se compone de las evaluaciones con diferentes métricas de rendimiento, realizadas sobre los diferentes modelos creados, con las características estudiadas en la actividad anterior.

A.2. Planificación temporal

La planificación temporal que se ha planteado en el proyecto la podemos observar en la Tabla A.1.

Actividad	**Duración**
Investigación	7 días
Preparación del entorno	7 días
Exploración de datos	14 días
Adaptación de datos	7 días
Pruebas iniciales	7 días
Estudio/Evaluación de parámetros	14 días
Elaboración/Evaluación modelos finales	14 días

Tabla A.1: Estimación de la duración de las actividades de la estancia en el GIR

Podemos destacar como las operaciones que más trabajo suponen son la exploración de datos, el estudio y evaluación de parámetros, y la elaboración/evaluación de modelos finales, debido a que tienen un peso mayor sobre el desarrollo, y debemos de estimar con el tiempo suficiente para que las tareas estén completadas antes de empezar con la siguiente, y comprobar que todos los objetivos de cada una se han cumplido al final de las mismas.

A.3. Estudio de viabilidad

Antes de realizar un proyecto, debemos de analizar si es posible su ejecución y desarrollo. Por ello, debemos de realizar un estudio sobre la viabilidad del mismo, indicando los motivos principales de por qué el desarrollo de este estudio es viable.

Debemos destacar que este estudio es una solución alternativa al trabajo realizado por J. Vegas y C. LLamas [7], en el que la hipótesis principal se centra en si la forma en que un usuario interactúa con una manija de puerta es adecuada para ser utilizada en la tarea de identificación, utilizando técnicas clásicas de Machine Learning. Afirmando esta hipótesis, obteniendo resultados favorables del área bajo la curva (AUC), con unos valores en torno a 0.90, podemos concluir en que existe la posibilidad de realizar una solución alternativa, implementando algoritmos y técnicas propias de Deep Learning, que es en lo que se centrará este estudio, con el objetivo principal de desarrollar un modo de identificación de individuos, mediante la interacción de los mismos con el dispositivo de apertura de una puerta.

Viabilidad económica

Respecto a la viabilidad económica del estudio, debemos decir que al tratarse de un trabajo donde se ha trabajado con datos e información ya adquiridos, y que estaban en poder del grupo de investigación, no ha sido necesario adquirir ningún elemento que suponga costes en el trabajo, además de que todos los recursos utilizados, tanto las herramientas de desarrollo como las librerías, son de código abierto y están accesibles para cualquier persona que quiera utilizarlos.

Apéndice B

Documentación del Desarrollador

En este apéndice se incluye la documentación necesaria para que el desarrollador de aplicaciones pueda comprender la estructura de la solución software aportada y poder modificarla lo que sea necesario para conseguir la solución al problema.

B.1. Introducción

El código del proyecto se encuentra en el repositorio de github `https://github.com/aitorojeda/DatosTFM.git`, organizado en 2 directorios diferentes: Ficheros de datos, donde podremos encontrar los datos que se han usado en el trabajo; y Ficheros de código, donde encontraremos los ficheros de código utilizados para el desarrollo del mismo.

Como ya se ha explicado en la Sección 4.1, el entorno que se ha utilizado para el desarrollo ha sido *Anaconda Navigator*. Se trata de un entorno local que nos permite explotar los recursos disponibles en nuestra máquina personal sin necesidad de recurrir a recursos de terceros o utilizar software en la nube, con todas las restricciones de recursos que eso supone. Además de dicho entorno, se utilizará conjuntamente Jupyter y Pyspark, que son las herramientas fundamentales para el desarrollo del trabajo, y con las que se han podido realizar todas las tareas asociadas al trabajo. A continuación, se explicarán los pasos de manera más detallada para poner a punto el entorno local, y comenzar a ejecutar el código desarrollado. Es importante que antes de realizar la instalación, este descargada e instalada la última versión de *Python*, ya que es el lenguaje que se ha utilizado en el desarrollo del trabajo.

Descarga e instalación del entorno

Para descargar el entorno, debemos de seguir los siguientes pasos:

- Lo primero que debemos de hacer es dirigirnos a nuestro navegador, y buscar "Descargar Anaconda Navigator" , donde haremos click en el primer enlace.
- Una vez dentro de la página, elegiremos el instalador apropiado para nuestro sistema operativo, (Windows, MAC, Linux), y descargaremos el instalador.
- Con el instalador ya descargado, lo abrimos, y seguimos los pasos y las instrucciones de la guía de instalación, para completar la instalación del entorno.

Lo primero que debemos de hacer es dirigirnos a nuestro navegador, y buscar " Descargar Anaconda Navigator" , donde haremos click en el primer enlace. Una vez dentro de la página, elegiremos el instalador apropiado para nuestro sistema operativo, (Windows, MAC, Linux), y descargaremos el instalador. Cuando éste se haya descargado, lo abriremos y seguiremos los diferentes pasos que nos aparecen en la guía para completar la instalación. Por último, si abrimos la aplicación, en la Figura B.1 podemos observar la imagen de la interfaz principal de la misma.

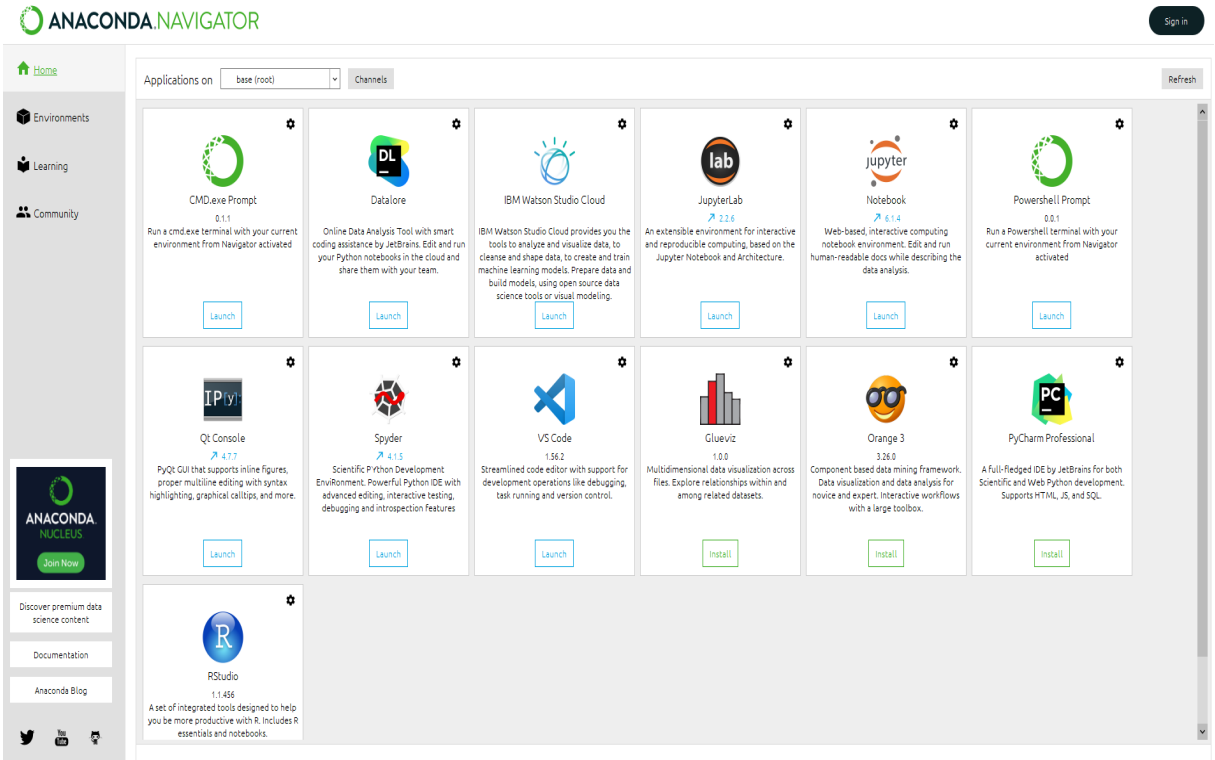

Figura B.1: Interfaz de inicio de la aplicación *Anaconda*.

Creación de un entorno de trabajo

Con el entorno ya descargado e instalado, debemos abrirlo y crear un entorno de trabajo, en vez de utilizar el entorno que aparece por defecto, debido a que podemos realizar diferentes tipos de trabajos, y tener en cada uno instaladas librerías diferentes. Para ello, debemos abrir Anaconda, pinchar en la sección "Environments", y hacer click en el icono de "Create". Nos aparecerá una pestaña nueva, en la que podremos elegir el Lenguaje (Python 3.8), y el nombre que le pondremos al entorno. Una vez hecho, le daremos a "Create" y esperaremos a que se instalen todas las dependencias necesarias en el nuevo entorno. Para finalizar, nos dirigiremos a la página de inicio de la herramienta, e instalaremos todos los módulos que vayamos a utilizar en el desarrollo del proyecto (Jupyter, Pycharm, Spyder, etc), en este caso, solo haremos la instalación de *Jupyter*, y al finalizar, ya podremos crear o modificar ficheros para comenzar a trabajar.

Instalación de los componentes necesarios

Para el desarrollo de este trabajo, ha sido necesario realizar la instalación de diversas librerías y herramientas propias de Python y de Deep Learning, por tanto, vamos a explicar los pasos necesarios y los comandos específicos para su instalación.

- Buscamos en nuestro equipo la consola de comandos de Anaconda: *Anaconda Prompt*, y la abrimos para introducir una serie de comandos.
- A continuación, elegimos el entorno creado: `conda activate <nombre del entorno>`
- Una vez dentro de nuestro entorno, instalamos las librerías *numpy, matplotlib, scipy, pandas* y *sklearn* utilizando el comando: `conda install <nombre librería>`
- Comprobamos que se han instalado utilizando el siguiente comando: `conda list`
- También, debemos instalar *Pyspark* para utilizar algunas funciones que incluye la herramienta, simplemente deberíamos instalar el paquete (pyspark) con el comando mostrado anteriormente.
- Por último, instalamos las librerías de Deep Learning necesarias, que son *tensorflow* y *keras*.

Una vez se han instalado todas las librerías y herramientas necesarias, ya se puede empezar a trabajar en el desarrollo del código.

B.2. Ejecución del código desarrollado

Una vez ya tenemos montado todo nuestro entorno de trabajo, para ejecutar el código, podemos realizarlo de 2 maneras diferentes:

- Ejecución desde Jupyter online. Para ejecutar el código sin necesidad de descargar ningún tipo de software, debemos de seguir los siguientes pasos:

  1 Buscar en el navegador "Jupyter notebook online" y hacer click en la primera página.

  2 Hacer click en el apartado "Try Jupyter Lab".

  3 Cuando se haya iniciado el repositorio, justo debajo de la barra de herramientas, hacer click en el icono "Upload Files", elegir el fichero de código que vayamos a utilizar y, una vez se haya cargado, ya podremos comenzar a editar y ejecutar el código.

- Ejecución desde Jupyter desde el entorno local. Como ya hemos configurado un entorno, podemos empezar a ejecutar el código de desarrollo siguiendo los siguientes pasos:

  1 Abrir anaconda navigator, y en la pestaña de "Environments", hacer click en el entorno que se ha creado para el desarrollo.

  2 Una vez cargado el entorno, elegiremos la herramienta *Jupyter* y esperaremos a que aparezca una ventana del navegador donde se haya cargado la herramienta.

  3 Con la herramienta ya cargada, elegiremos el fichero a ejecutar de nuestra estructura de ficheros local, y ya podremos consultar, modificar, y ejecutar por módulos (o de manera completa) el código de desarrollo.

Bibliografía

[1] *Tiefenau, C.; Häring, M.; Khamis, M.; von Zezschwitz, E. Please enter your PIN—On the Risk of Bypass Attacks on Biometric Authentication on Mobile Devices*

[2] *Zhu, T.;Weng, Z.; Chen, G.; Fu, L. A Hybrid Deep Learning System for Real-World Mobile User Authentication Using Motion Sensors.*

[3] *Abuhamad, M.; AbuHmed, T.; Mohaisen, D.; Nyang, D.H. AUToSen: Deep-Learning-Based Implicit Continuous Authentication Using Smartphone Sensors.*

[4] *Lee,W.; Lee, R. Implicit Sensor-Based Authentication of Smartphone Users with Smartwatch. In Proceedings of the Hardware and Architectural Support for Security and Privacy 2016, Seoul, Korea, 18 June 2016*

[5] *Choi, M.; Lee, S.; Jo, M.; Shin, J. Keystroke Dynamics-Based Authentication Using Unique Keypad*

[6] *Lamiche, I.; Bin, G.; Jing, Y.; Yu, Z.; Hadid, A. A continuous smartphone authentication method based on gait patterns and keystroke dynamics.J. Ambient. Intell. Humaniz. Comput. 2018.*

[7] *Jesús Vegas, César Llamas, Manuel A. González; Carmen Hernández.Identifying users from the interaction with a door handle.Pervasive and Mobile Computing 70.2021.*

[8] *Joseph Romanowski, Kirsanov Charles, Patricia Jasso, Shreyansh Shah, and Hugh W Eng. 2016. A Biometric Security Acceptability and Ease-of-Use Study on a Palm Vein Scanner. Proceedings of Student-Faculty Research Day, CSIS, Pace University (2016).*

[9] *Fujitsu. 2018. Fujitsu Begins Large-Scale Internal Deployment of Palm Vein Authentication to Accelerate Workstyle Transformation. Fujitsu Press Release (2018). http://www.fujitsu.com/global/about/resources/news/ press-releases/2018/0118-01.html.*

[10] *K. Yeh, C. Su, W. Chiu and L. Zhou, "I Walk, Therefore I Am: Continuous User Authentication with Plantar Biometrics," in IEEE Communications Magazine, vol. 56, no. 2, pp. 150-157, Feb. 2018, doi: 10.1109/MCOM.2018.1700339.*

[11] *V. Yano et al., "Extraction and Application of Dynamic Pupillometry Featuresfor Biometric Authentication," Measurement, vol. 63, 2015, pp. 41–48*

[12] *K.Annapurani et al., "Fusion of Shape of the Ear and Tragus — A Unique Feature Extractionmethod for Ear Authentication System" Expert Systems with Applications, vol. 42, issue 1, Jan. 2015, pp. 649–656.*

[13] *S. Sarkar et al., "Deep Feature-Based Face Detection on Mobile Devices," Proc. IEEEInt'l. Conf. Identity, Security and Behavior Anal., 2016.*

[14] *N. Reddy, A. Rattani and R. Derakhshani, "Comparison of Deep Learning Models for Biometric-based Mobile User Authentication," 2018 IEEE 9th International Conference on Biometrics Theory, Applications and Systems (BTAS), 2018, pp. 1-6, doi: 10.1109/BTAS.2018.8698586.*

[15] *I. Chamatidis, A. Katsika and G. Spathoulas, "Using deep learning neural networks for ECG based authentication," 2017 International Carnahan Conference on Security Technology (ICCST), 2017, pp. 1-6, doi: 10.1109/CCST.2017.8167816.*

[16] *K. Fujinami, S. Pirttikangas, T. Nakajima, Who opened the door?: Towards the implicit user identification for sentient artefacts, in: International Conference on Pervasive Computing, Vol. 207, 2006, pp. 107–111.*

[17] *R. Piltaver, H. Gjoreski, M. Gams, Identifying a person with door-mounted accelerometer, J. Ambient Intell. Smart Environ. 10 (5) (2018) 361–375.*

[18] *Volaka, H.C.; Alptekin, G.; Basar, O.E.; Isbilen, M.; Incel, O.D. Towards Continuous Authentication on Mobile Phones using Deep Learning Models. Procedia Comput. Sci. 2019, 155, 177–184.*

[19] *Goodfellow, I.; Bengio, Y.; Courville, A. Deep Learning; MIT Press: Cambridge, MA, USA, 2016; Volume 1, p. 692*

[20] *Bengio, Y.; Courville, A.; Vincent, P. Representation Learning: A Review and New Perspectives. IEEE Trans. Pattern Anal. Mach. Intell. 2013, 35, 1798–1828*

[21] *Bengio, Yoshua; Lee, Dong-Hyun; Bornschein, Jorg; Mesnard, Thomas; Lin, Zhouhan (13 February 2015). "Towards Biologically Plausible Deep Learning". arXiv:1502.04156*

[22] *Krizhevsky, Alex; Sutskever, Ilya; Hinton, Geoffry (2012). "ImageNet Classification with Deep Convolutional Neural Networks". NIPS 2012: Neural Information Processing Systems, Lake Tahoe, Nevada.*

[23] *Li, Xiangang; Wu, Xihong (2014-10-15). Çonstructing Long Short-Term Memory based Deep Recurrent Neural Networks for Large Vocabulary Speech Recognition"*

[24] *Huang, Jie; Zhou, Wengang; Zhang, Qilin; Li, Houqiang; Li, Weiping (2018-01-30). "Video-based Sign Language Recognition without Temporal Segmentation*

[25] *Ravanelli, Mirco; Brakel, Philemon; Omologo, Maurizio; Bengio, Yoshua (2018). "Light Gated Recurrent Units for Speech Recognition". IEEE Transactions on Emerging Topics in Computational Intelligence. 2 (2): 92–102*

[26] *Chung, Junyoung; Gulcehre, Caglar; Cho, KyungHyun; Bengio, Yoshua (2014). Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modelin.*

[27] *Best Python Libraries for Machine Learning. GeeksForGeeks.* Recuperado de *https://www.geeksforgeeks.org/best-python-libraries-for-machine-learning/.* Última fecha de consulta: 15/08/2021

[28] *Jason Brownlee. Deep learning with python. Machine Learning Mistery.* Recuperado de *https://machinelearningmastery.com/.* Última fecha de consulta: 23/08/2021

[29] *François Chollet. Deep Learning with Python. 2017. 1617294438. Manning Publications Co., USA. 1st Edition*

[30] *Pavan Vadapalli. Deep Learning Techniques You Should Know About.* Recuperado de *https://www.upgrad.com/blog/top-deep-learning-techniques-you-should-know-about.* Última fecha de consulta: 15/08/2021

[31] *Wikipedia. Deep Learning.* Recuperado de *https://en.wikipedia.org/wiki/Deep_learning.* Última fecha de consulta: 14/07/2021

[32] *S. Sarkar et al., "Deep Feature-Based Face Detection on Mobile Devices," Proc. IEEE Int'l. Conf. Identity, Security and Behavior Anal., 2016.*

[33] *Deep Learning Frameworks. UpGrad Blog. https://www.upgrad.com/blog/top-deep-learning-frameworks/#6_Swift_for_TensorFlow.* Última fecha de consulta: 01/09/2021

[34] *Long short-term memory. Wikipedia.* Recuperado de *https://en.wikipedia.org/wiki/Long_short-term_memory.* Última fecha de consulta: 16/07/2021

[35] *Choi, K.; Ryu, H.; Kim, J. Deep Residual Networks for User Authentication via Hand-Object Manipulations. Sensors 2021, 21, 2981. https://doi.org/10.3390/s21092981*

[36] *Deep Learning Environments. CodeAcademy. https://www.codecademy.com/articles/setup-deep-learning-environment.* Última fecha de consulta: 02/09/2021

[37] *Deep Learning Frameworks. SimpliLearn. https://www.simplilearn.com/tutorials/deep-learning-tutorial/deep-learning-frameworks.* Última fecha de consulta: 02/09/2021

[38] *Google Collaboratory. Google Research. https://research.google.com/colaboratory/-intl/es/faq.html.* Última fecha de consulta: 07/09/2021

[39] *What is Machine Learning?. Datarevenue. https://www.datarevenue.com/en-blog/what-is-machine-learning-a-visual-explanation.* Última fecha de consulta: 18/08/2021

[40] *What is Deep Learning?. Machine Learning Mistery. https://machinelearningmastery.com/what-is-deep-learning/.* Última fecha de consulta: 04/08/2021

[41] *Choi, K.; Ryu, H.; Kim, J. Deep Residual Networks for User Authentication via Hand-Object Manipulations. Sensors 2021, 21, 2981. https://doi.org/10.3390/s21092981*

[42] *Install Anaconda on Windows. https://docs.anaconda.com/anaconda/install/windows/.* Última fecha de consulta: 15/09/2021

[43] *Entorno de trabajo Anaconda. https://www.aprendemachinelearning.com/instalar-ambiente-de-desarrollo-python-anaconda-para-aprendizaje-automatico/.* Última fecha de consulta: 15/09/2021