Universidad Miguel Hernández

Facultad de Ciencias Sociales y Jurídicas

Grado en Estadística Empresarial

Trabajo de Fin de Grado

Curso 2021-2022

Gamificación con Azure Kinect para la mejora de la condición física en personas de la tercera edad:

Uso de la inteligencia artificial en vídeos

Autora: Lidia Cecilia Madera Guerrón

Tutora: María Asunción Martínez Mayoral

ÍNDICE

1. Agradecimientos

En primer lugar, me gustaría agradecer a la empresa Instead Technologies por entregar toda su confianza y apoyo en este proyecto tanto a mis compañeros como a mí. Su flexibilidad y profesionalidad hicieron un gran cimiento para poder llevar a cabo la idea principal del trabajo. Así pudo nacer la idea de la gamificación que se espera en un futuro darle vida para que pueda ayudar a las personas mayores o en proceso de rehabilitación.

Por otra parte, a mi equipo, agradecer todo el esfuerzo y cooperación que me han brindado durante la trayectoria del trabajo de fin de grado. El proceso fue de lo más gratificante y me otorgó un conocimiento extra sobre las habilidades de los otros grados participantes.

Y, por último, a mi familia. Ellos fueron mi pilar fundamental para poder avanzar con fuerza y actitud estos años de carrera. Me hicieron valorar y disfrutar de esto tanto académica como personalmente.

2. Resumen

La inteligencia artificial es una vía donde se aplica una o varias habilidades humanas a una máquina con la intención de que esta realice acciones de forma automática, siendo capaz de reproducir de forma similar como lo haría un humano. La IA, en la actualidad, está en un estado exponencial desde hace unos años, y uno de los motivos por los que se está ampliando su uso en multitud de usuarios es para mejorar o cambiar sus vidas.

Los grupos que engloba la Inteligencia Artificial es el Machine Learning y dentro de este es el Deep Learning. Para este proyecto nos centraremos en Deep Learning, lo que es conocido como también "Aprendizaje Profundo". Este procedimiento se aplica para que los datos, que se generan en cualquier ámbito, sean estudiados de una manera natural por medio de ordenadores. Más adelante, explicaremos y pondremos ejemplos de los usos del Deep Learning en la vida real, sin embargo, nos enfocaremos en la técnica de captación de patrones en imágenes o videos, percibiendo desde objetos fijos hasta personas en movimiento. Con el paso del tiempo se han ido mejorando estos procesos para que la detección de objetivos sea de los más asertiva y con un porcentaje de error mínimo.

Con este concepto inicial se busca indicar que el objetivo principal de este proyecto es la predicción de movimientos en personas mayores cuando ejecutan una serie de ejercicios físicos. Esta es una forma de orientar al usuario de cuáles son los movimientos que debe realizar, y según su condición física, la meta es corregir esas deficiencias. Ya que se pretende evitar enfermedades a largo plazo que son generadas por el senderismo, la edad, malos hábitos, etc.

Se contó, durante el desarrollo del plan, con la ayuda y las herramientas de la empresa Instead Technologies. La idea inicial de la empresa era emplear el mecanismo Azure Kinect, con la finalidad de iniciar un prototipo en el ámbito de la salud. Azure Kinect es una herramienta capaz de captar los puntos principales del cuerpo en los usuarios a la hora de ponerse en frente de esta. Por lo que, nuestra intención era buscar una unión entre un juego y una planificación de movimientos con aumento de eficiencia motriz para personas de la tercera edad. Se buscaba envolver al paciente en un entorno virtual con el fin de que no sienta que está entrenando, sino que sirva como motivación para reproducir los movimientos diseñados.

Este TFG pasó a ser un TFGi (trabajo de fin de grado interdisciplinar) por la colaboración de dos grados universitarios debido a que se pretendía fusionar las tres carreras universitarias (Terapia Ocupacional, Fisioterapia y Estadística Empresarial) para alcanzar una idea de conceptos que ofrezcan un prototipo capaz de llevarlo a la vida real llamado "*Aging Rehab"*. Con esta idea se pretendía realizar un videojuego basado en ejemplos que se encuentran en el mercado actual, pero aplicando novedades para ser adaptados en centros de salud.

En definitiva, la idea es juntar todos los conceptos y movimientos en un único planteamiento que hoy en día es algo muy conocido como la gamificación. Este término está abarcando a más campos como, por ejemplo: finanzas, deportes, educación, marketing, etc. La gamificación consiste en unas técnicas de aprendizaje con motivaciones, recompensas y objetivos. Es una vía que se está digitalizando y se puede encontrar tanto en ordenadores como en móviles. Nuestra idea es que comience

computándose con ayuda de un médico evaluando y comunicando al usuario. Y más adelante, una forma más económica y flexible sería llevarlo a una aplicación para el móvil. Esto mejoraría su uso ya que podría emplearse desde casa. Previamente se comprobaría que no se corra ningún riesgo ya que se usaría sin supervisión.

Los métodos que se usaron por parte de los alumnos y la empresa fueron dirigidos en la mejora de la condición física, uso de inteligencia artificial y planificar ejercicios para mejorar la postura de los usuarios. Con esto, se abarcarían varios sectores para que la idea fuese de lo más sólida y tuviese diferentes perspectivas tanto académicas como personales.

Se consiguió realizar tres movimientos con la Azure Kinect para comprobar si era capaz de detectarse ante la cámara sin solaparse y provocar error. Por lo que nos basamos en esas acciones para proceder a la lectura de datos y ejecutarlas en el código creado por la alumna de Estadistica Empresarial.

Por último, se pretende que una vez terminado el código generado correctamente nos ayude a predecir qué tipo de movimiento está realizando el paciente y poder diagnosticar si su movilidad es funcional o no. Esto se realizará mediante redes neuronales de clasificación binaria. Al final de este trabajo se podrá corroborar con datos si el sistema que se empleó durante todo el proyecto es adecuado para avanzar con el prototipo y comprobar cuales son las deficiencias en un futuro.

Palabras claves: Inteligencia Artificial, Machine Learning, Deep Learning, Redes neuronales, Predicción, Análisis de videos, Clasificación binaria, Azure Kinect, Movimientos funcionales, TFGi, Prototipo, Planificación de movimientos, Fisioterapia, Terapia Ocupacional, Gamificación.

3. Introducción

3.1 Motivación

El TFG se ha realizado en la modalidad de Programa Interdisciplinario de Fin de Grado (TFGi). Este tipo de trabajo de fin de grado se deriva del plan de innovación docente y se compone de un proyecto final de carrera realizado conjuntamente por un equipo de alumnos de diferentes titulaciones.

Cada alumno es tutorizado por un profesor con la temática de que se trata y además de un tutor asociado por el Programa Interdisciplinar para poder dirigir la evolución del trabajo desde distintos prismas. Dicho TFGi propone una solución global para la resolución de problemas y desafíos prácticos de proyectos de empresas. Lo que significa que se considera una fusión entre los conocimientos profesionales y los adquiridos durante la trayectoria académica.

Por tanto, el contenido de dicho trabajo no solo incluye contextos, objetivos, métodos, resultados y conclusiones correspondientes a las habilidades y conocimientos del estudiante de la titulación que lo presenta, sino que también el proceso que demostró el equipo interdisciplinario.

En concreto, se hallarán evoluciones con respecto al proyecto, tanto de forma generalizada como de los participantes de las otras titulaciones por separado. Ya que por naturaleza del trabajo estos aspectos no se pueden separar de sí mismos para que mantenga su esencia.

En este TFGi colaboraron los estudiantes de la Universidad Miguel Hernández: Iván Pacheco del grado de Fisioterapia, Leticia Paz del grado de Terapia Ocupacional y Lidia Madera del grado de Estadística Empresarial que abordaron la temática de la movilidad y daño corporal con referencia a personas de tercera edad. Se unieron estos estudios para poder programar tratamientos progresivos, valorar su funcionalidad y clasificar los movimientos en grupos eficientes o deficientes.

En conclusión, tras la agrupación del equipo interdisciplinar, se inició la comunicación con la empresa Instead Technologies que ofreció, en una de las reuniones del Programa Interdisciplinar, su instrumental y conocimientos referentes al asunto de la ciencia y la salud, para lograr una idea que vincule ambos conceptos puesto que son el lema principal de la compañía.

Por lo que, realizando una lluvia de ideas del equipo, se sugirió a la empresa realizar un concepto dinámico y sencillo que puede llegar a mejorar la condición física y mental de personas de la tercera edad. La idea fue desarrollar un diseño de juego con el dispositivo Azure Kinect, el cual nos proporcionó la empresa, con la finalidad de potenciar y evaluar los movimientos de forma progresiva de dichas personas realizando tareas básicas para un posterior análisis.

Hoy en día, la tercera edad es una comunidad, que con que el paso del tiempo, comienzan a perder fuerza y motivación para la realización de actividades físicas. Conllevando a las limitaciones de las actividades básicas e instrumentales de la vida diaria. Así centrándonos en ese criterio, por parte de los grados de Fisioterapia y Terapia Ocupacional se enfocaron en labores habituales como el peinarse, el comer, el vestirse como acciones que implique una leve movilidad para encontrar patrones. Y por parte del grado de Estadística Empresarial se comprometió en estudiar dichos movimientos y analizarlos mediante los datos obtenidos del programa generado por la empresa. Se realizó una lectura y análisis de los datos con la compilación de código relacionado con el Deep Learning mediante redes neuronales.

Este conjunto de procesos daba la resolución a si la movilidad del usuario era eficiente o deficiente en comparación a los movimientos base que fueron creados por el equipo interdisciplinar.

3.2 Contexto de redes neuronales

Lo que hoy conocemos como redes artificiales, es un tema que se ha estudiado desde inicios del siglo XX. Sin embargo, desde los últimos años es cuando ha empezado a crearse el "boom". Por tanto, estas han sido usadas tanto académicamente como de investigación con el paso del tiempo debido a su evolución y mejora. Algunos ejemplos de ellos son: AlphaGo, Facebook, Alexa…

Concepto de las redes neuronales:

En la actualidad, las redes neuronales están en todo su esplendor ya que son una serie de modelos caracterizados por poseer gran espacio de parámetros y una estructura flexible. Su función consiste en almacenar conocimiento experimental de manera natural y ponerlo a disposición para su uso.

El proceso se centrará en la aplicación de redes neuronales, por ello, se explicará de forma detalla el concepto de red neuronal.

La arquitectura de las redes es similar a la del cerebro humano. La red recibe una serie de valores de entrada y se colocan a modo de nodo, a la cual recibe el nombre de neurona. Estas se unen secuencialmente en capas. Entonces, en el caso de las redes neuronales se les adjudica un peso y valor numérico para modificar el valor de entrada. Este proceso se considera un tipo de aprendizaje automático "Machine Learning".

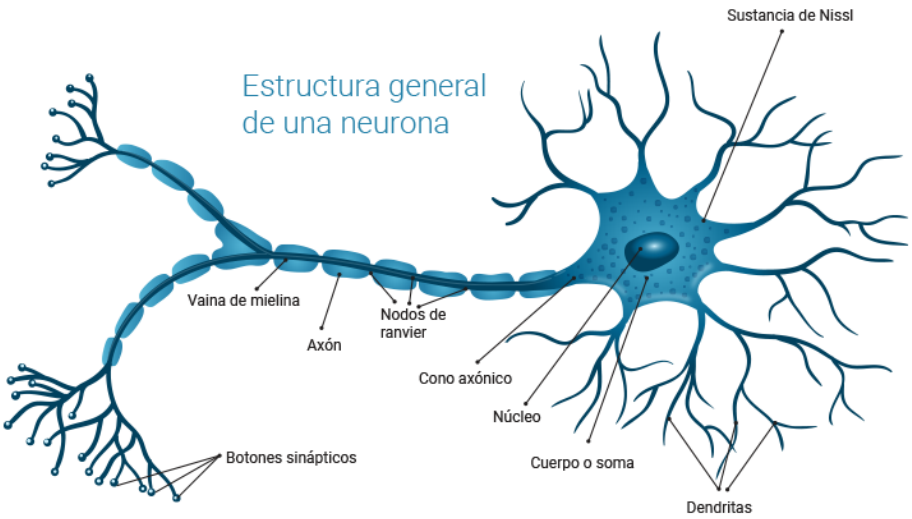


Figura 1: Imagen de la estructura de la red humana

Fuente: (Campos Soberanis, 2018). [2]

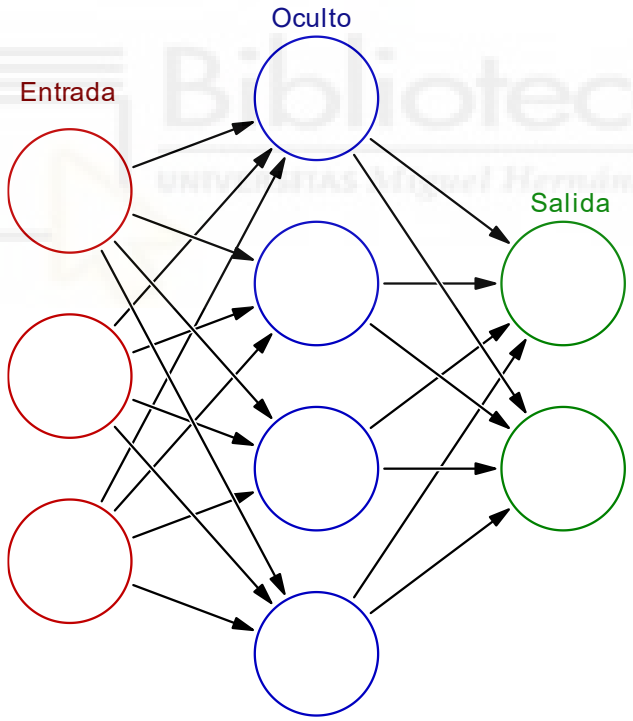


Figura 2: Gráfico de la estructura de las redes neuronales

Fuente: (en:User:Cburnett, 2019). [4]

Para entrar más en detalle, lo que nos ayudará a interpretar el uso de estos será el Deep Learning "Aprendizaje Profundo". Su procedencia viene de la inteligencia artificial que deseaba mejorar la técnica de clasificar, detectar y detallar. Existen variedad de tipos de aprendizaje: supervisado, no supervisado y de refuerzo.

Esto es lo que consiste cada aprendizaje (Jorge Matich, 2011, pág. 19) [7]:

- **Aprendizaje supervisado**: Es el estudio basado en tareas. Consiste en que los datos usados en los algoritmos son revisados y organizados por el humano para indicar que tipo de salida se busca.
- **Aprendizaje no supervisado**: Este tipo de aprendizaje reside en datos. Aquí no participa el humano y tampoco se le indica que conclusión se indaga.
- **Aprendizaje de refuerzo**: Aquí se aprende según la reacción al entorno el cual se estudia. Dicha instrucción se va reforzando positivamente cuando se acierta a la hora de sacar la solución. Todo en base a la experiencia.

El tipo de aprendizaje, en el cual nos centraremos, será el supervisado. Se trata de perseguir la idea de reconocimiento de patrones, llamado como clasificación, al cual se les aplican datos secuenciales y que se manipulan empleando tácticas basadas en las matemáticas.

Estas se han convertido en la herramienta de elección para muchas aplicaciones de minería de datos predictivas ya que tienen potentes funciones, fácil uso y flexibilidad.

Por lo tanto, este fue el procedimiento clave que se implementó en el proceso para el análisis de videos recogidos mediante el aparato Azure Kinect.

Historia de las redes neuronales y su evolución (Jorge Matich, 2011, pág. 6) [7]:

- 1936: **Alan Turing,** fue el primero en centrarse en el estudio del cerebro enfocándose de forma computacional.

- 1943: **Warren McCulloch y Walter Pitts**, un neurofisiólogo y un matemático fueron los que iniciaron una teoría acerca del funcionamiento de las neuronas. Estos crearon un modelo matemático al cual llamaron "lógica umbral".

- 1949: **Donald Hebb**, un psicólogo que realizó su propia hipótesis sobre las secuencias de aprendizaje de una red neuronal considerándose "aprendizaje no supervisado". Él reflexionó acerca del aprendizaje, puesto que, según él este proceso se producía por unos determinados cambios que activaban las neuronas.

- 1954: **Farley y Wesley A. Clark**, estos iniciaron el uso de máquinas computacionales para aplicar el modelo de redes de Hebb.

- 1958: **Frank Rosenblatt**, dio paso a la Perceptrón. Que se conoce como una red neuronal que reconoce patrones, es la más antigua. Esta es capaz de

reconocer patrones similares, aunque anteriormente no se les hubiese indicado. Por ello es la más usada pese a sus limitaciones.

- 1960: **Bernard Widrow y Marcial Hoff**, estos desarrollaron el modelo Adaline, que viene de "Adaptative Linear Elements". Este prototipo se implementó de un ejemplo de la vida real. El problema en el que se usó fue para descartar los ecos producidos en las llamadas telefónicas.

- 1965: **Ivakhnenko**, realizó un sistema de unión para manejar datos con redes funcionales multicapas.

- 1975: **Werbos**, pudo resolver de manera vigorosa un problema mediante un algoritmo de propagación hacia atrás con redes de multicapas. El proceso de propagación consiste en hallar la diferencia del resultado que se obtiene al que se pretende adquirir con diversas transformaciones entre los pesos de las redes.

- De **1992** hacia **2010**: se han ido alcanzando progresivamente una popularidad y en 1992 se incorporó el max-pooling. Esto consiste en un submuestreo en el cual la información se divide en grupos de tamaño similar para dar un valor máximo a cada uno de ellos y ayudar a identificar los objetivos tridimensionales. Por lo tanto, con el paso del tiempo en 2010 el uso del max-pooling se agilizó por los GPUs y se pudo indicar que este tipo de rendimiento genera mejores resultados.

Principales usos en la actualidad de las redes (Gómez Gil, 2016) [5]:

- Reconocimiento de manuscritos
- Reconocimiento de voz y facial
- Vehículos de conducción autónoma
- Clasificación de objetos en imágenes y videos
- Análisis de comportamiento de ventas y productividad
- Investigación en medicina

3.3 Introducción de Azure Kinect

Iniciaremos el apartado comentando que es Azure Kinect y sus valoraciones durante el uso en este proyecto.

Concepto:

Azure Kinect es un kit con sensores de inteligencia artificial el cual les genera modelos de profundidad y de seguimiento corporal. El control de colores y sensores de movimiento permiten que la sincronización del dispositivo con relación a otros durante la realización del desplazamiento sea de carácter virtuoso.

Este, además, nos puede separar cada esqueleto anatómicamente si se encuentran varios cuerpos ante la cámara.

Ventajas:

Se expondrán los motivos por los cuales se usó Azure Kinect y no otro tipo de dispositivo u otras versiones de Kinect.

- **Mejora en profundidad**:

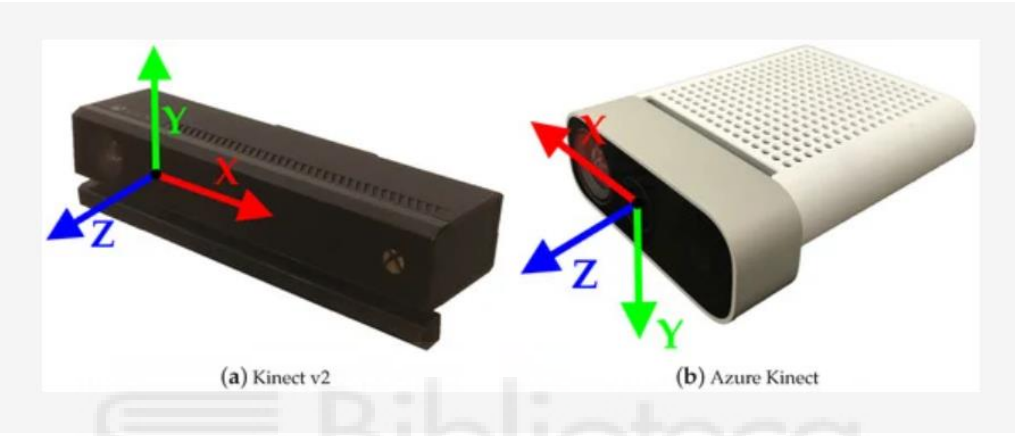
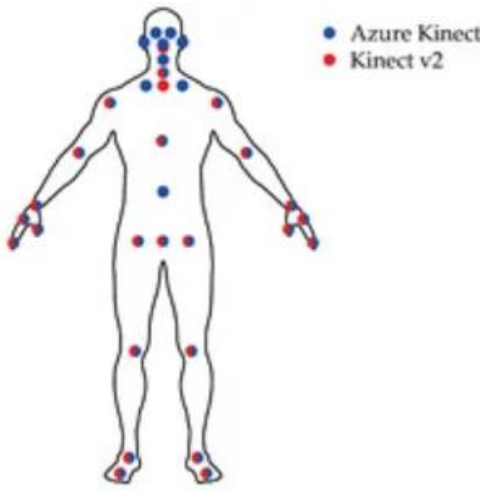


Figura 3: Comparación Kinect V2 con Azure Kinect

Fuente: (Amadeus Albert, y otros, 2020) [1]

Por experiencia de la empresa con los dispositivos y corroborando con la documentación encontrada se notó como la captación de imágenes fue de lo más cercana a la realidad. Puesto que se seguía percibiendo todos los movimientos y giros en la cámara sin ningún problema de lectura y con una tasa de error mínima.

- **Aumento de puntos detectables**:



(b) Kinect v2 and Azure Kinect skeletons

Figura 4: Captura de nodos con Kinect V2 y Azure Kinect

Fuente: (Amadeus Albert, y otros, 2020) [1]

Describiendo la fotografía podemos mencionar que la Azure Kinect proporciona una representación más detallada del cuerpo humano ya que capta en detalle los tramos desde cabeza-hombro-cadera, que son las partes de interés para el estudio.

En las imágenes anteriores se comprueba como con la Azure Kinect fue todo un acierto en el uso de este trabajo.

Aplicación dentro del proyecto:

Tras estos conceptos, observaremos en la imagen cuales son los puntos que capta la Kinect con el programa generado desde Python. La precisión de captura que se generaba en la Kinect era de las más alta en comparación a las anteriores versiones.

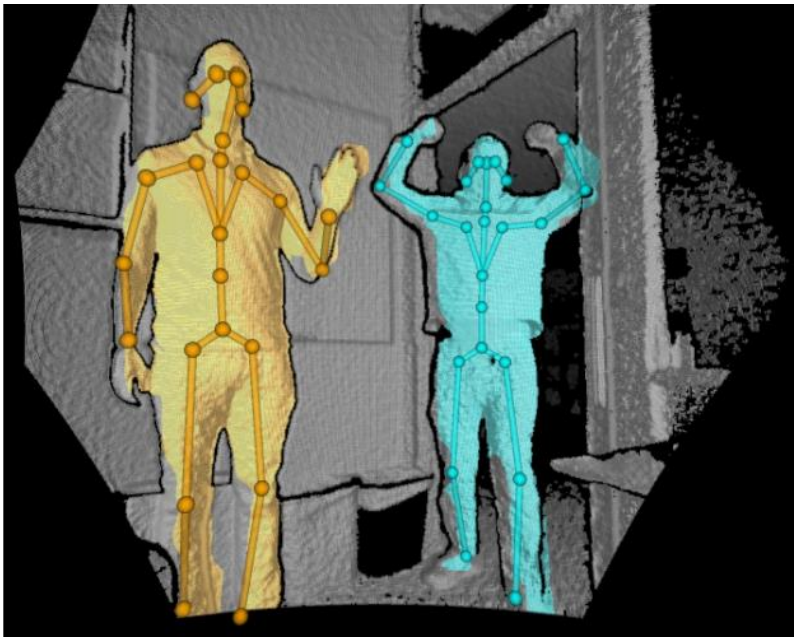


Figura 5: Puntos de captación con Azure Kinect

Fuente: (Microsoft, 2021, pág. 14) [8]

Después de diversas pruebas, se vio que se aceptaban leves giros del cuerpo, y aun así seguía detectando los mismos puntos de la parte del cuerpo sin que se solapen y pueda dar a un error de localización. Sin embargo, donde si se encontró algo de dificultad es en la habilidad motriz fina como por ejemplo la mano. Dentro de la mano no se podía detectar bien los dedos, sino que se detectaba como un solo punto. Por lo que nuestros movimientos fueron pensados en base a lo anterior para que sea sencillo a la hora de analizar los datos como para el individuo que reproduzca el movimiento en concreto.

Con cada prueba los alumnos de Terapia Ocupacional y Fisioterapia iban indicando cual era el movimiento más apropiado. En base a esas características se decidió cuáles deberían ser los grados de ángulos a los que se debe llegar, para que la persona que los desarrolle pueda realizar su movimiento de manera correcta (funcional).

La coordinación y conocimientos relacionado con la salud de ambos alumnos hicieron que más adelante se pudiera idear una serie de juegos estilo yincana para hacer un proyecto más dinámico y diferente a lo que se está habitualmente acostumbrado. El propósito es ir aplicando los juegos de forma secuencial para poder controlar la capacidad del paciente conforme al movimiento. Se aumentaría el nivel de dificultad si el paciente responde de manera conveniente. Y sino, minorar el nivel de esfuerzo si este no llegaba a conseguir el movimiento recomendado. Esta idea será desarrollada en los siguientes apartados de forma detallada comentando los juegos y sus reglas.

3.4 Herramientas usadas

En el presente proyecto, veremos que se usarán redes neuronales, y en concreto las de clasificación binaria con la finalidad de obtener dos grupos en las cuales se clasificarán en dos tipos. Los de tipo 0 los que son movimientos eficientes y tipo 1 los que son de movimientos deficientes.

Para ello se desarrolló un algoritmo en lenguaje de programación R tras ser entrenado para una base de datos seleccionada previamente por el equipo. Por lo que explicaremos cuales fueron los softwares que se emplearon durante todo el proyecto.

Softwares usados:

- **RStudio**: es un entorno de desarrollo de lenguajes que, en este caso, es de programación en R. Que dedicaremos exclusivamente a la estadística y gráficas. Su disponibilidad esta para Windows, Mac y Linux. Versión empleada R 4.1.2.

- **Google Drive**: su servicio consiste en almacenar archivos para poder mantenerlos de manera segura. También, permite el uso compartido y la edición instantánea para todos. En caso de que falle el ordenador, los documentos y las últimas modificaciones quedarán en la nube. Se usó entre los alumnos para facilitar la agrupación de datos durante el proyecto.

- **Base de datos**: la obtención de la BBDD será proporcionada por la empresa mediante algoritmos realizados en el lenguaje de programación Python, el cual captará las posiciones de las partes del cuerpo con ayuda de Azure Kinect. Los datos recogidos se guardaron en documentos con extensión en formato texto.

- **Anaconda**: Un entorno donde se encuentra el lenguaje de Python para el uso de ciencia de datos y aprendizaje automático. Se uso para conectar R y Python para la ejecución de Tensorflow y Keras. Se necesitó la actualización 2.1.0 de Tensorflow y 2.3.1 de Keras.

4. Objetivos

En este apartado se comentará los objetivos en común contextualizando en base al entorno de lo que busca la empresa y el grupo con su correspondiente justificación. Además, por la otra parte, en objetivos específicos se relacionarán las metas que busca el grado de Estadística con referencia al tema.

Objetivos generales:

Tras generar la idea en equipo, nuestro principal enfoque fue buscar una lista de movimientos funcionales, en base a las acciones que practican a diario. Con el fin de poder mejorar su condición o mantenerla si esta es buena.

Después, viendo que el listado de movimientos que se escogió fue el siguiente:

- Coger un plato y colocarlo en la estantería.
- Comer sopa y llevar la cuchara a la boca.
- Peinarse.

Se pretendía valorar la movilidad desde la parte central del cuerpo hasta la cabeza, comprobando cuales eran los puntos detectables en la Kinect. Para ello cogimos diferentes individuos para reproducir los movimientos. Cada uno realizó 6 repeticiones de manera correcta y 6 de forma incorrecta para la base de datos. Para no tener problemas en un futuro con los datos se hizo esto debido a que, si una misma persona o pocas personas realizan estos movimientos, podríamos tener errores aleatorios de repetitividad como de reproducibilidad.

- **Repetitividad**: es la variación de las mediciones que se extrajeron muchas veces de un mismo usuario con el mismo objeto, ambiente y manera de realizar la acción. Llamado como VE (variación de equipo).

Figura 6: Gráfico de Repetitividad

Fuente: (Infasdev, 2018) [6]

- **Reproducibilidad**: Se conoce como VO (variación de los operadores) y se produce por la variación del promedio de las medidas que se elaboran por distintos sujetos manejando el mismo sistema de medición.

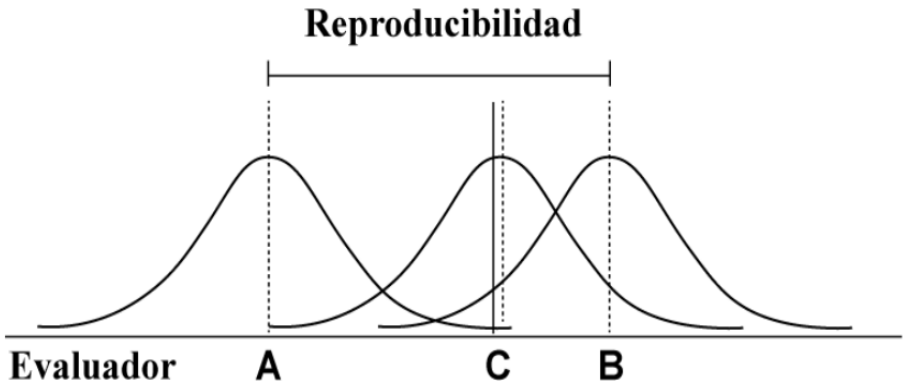


Figura 8: Gráfico de Reproducibilidad

Fuente: (Infasdev, 2018) [6]

Objetivos específicos:

Para continuar con el proyecto de la yincana y los movimientos base, la alumna del grado de Estadística inició con el uso de la inteligencia artificial para la identificación de movimientos funcionales y no funcionales en vídeos. Detalles a les cuales se centró:

- **Revisión bibliográfica sobre inteligencia artificial**: Principalmente se enfocó en la búsqueda de captación de videos y uso del Deep Learning. Recogida de información tanto en foros, webs académicas y libros relacionados.
- **Leer los datos numéricos y recopilar los que serán base de estudio:** Los datos extraídos con el software Python que empleó la empresa Instead Technologies. Verificar cuales eran los puntos necesarios para el estudio tras los movimientos de prueba que se hicieron en la empresa.
- **Poner en práctica algoritmos utilizando librerías de redes neuronales como Keras y Tensorflow para la predicción de movimientos captados con Azure Kinect**: Crear una algoritmia para la lectura de datos basada y uso de redes neuronales con el software Rstudio. Búsqueda de información sobre como usar Rstudio unido a Python para la ejecución de Tensorflow y Keras.

5. Metodología

Se relatará la metodología colaborativa global del equipo descomponiendo el problema y distribuyendo las tareas. Después, nos centraremos en procedimiento que se siguió de manera particular.

Metodología general:

Como inicio de partida los alumnos se rigieron a búsquedas tanto individuales como grupales. En este apartado veremos que hay una mezcla ya que es fundamental que haya una unión de las tres carreras que son desde un inicio de distintas ramas. Pondremos al final de la explicación un esquema de como fue el proceso de cada alumno y como se fue dando composición a la idea principal.

En primer lugar, la carrera de Fisioterapia y Terapia Ocupacional como en cierto modo si tuvieron relación en dicho proyecto, veremos como las búsquedas toman rutas similares.

Comenzando por la parte de Terapia Ocupacional, las búsquedas principales fueron una revisión bibliográfica de funcionalidades físicas y planificaciones de ejercicios que mejoren las funcionalidades motoras en personas de la tercera edad. Aportar ejemplos de propuestas de movimientos que se lleven a cabo en centros de rehabilitación o centros de mayores.

Después, la parte de Fisioterapia exploró páginas relacionadas con contenido de salud y, dentro de ella, la movilidad y evaluación del físico en personas de tercera edad. Ideó propuestas para planificar ejercicios que mejoren la movilidad o corrijan daños físicos en personas mayores. Siempre y cuando dichos pacientes no tengan ninguna patología diagnosticada.

Para concluir, en conjunto con todas las examinaciones en particular. La única en la que se centraron los tres alumnos fue en la cámara Azure Kinect. Se investigó en base a que es, que ofrece, y sus usos con relación al ámbito de la salud. Sin embargo, tras la búsqueda de información se observó que no hay mucho contenido sobre la Kinect a la hora de usarlo como método de evaluación física en personas mayores. Por lo que nuestro testimonio se basa en sus características principales y los datos recogidos después de las pruebas.

En definitiva, se buscó sobre la Azure Kinect las funciones principales y sus usos en la vida real. Así que se podría decir que su uso va aumentando poco a poco en el mundo de la salud. Y un ejemplo de ello sería la misma empresa que colaboró con el proyecto, que es Instead Technologies y lo aplica en este caso:

- **Nemty** (Instead Technologies)][9]: Es un diseño de análisis del comportamiento de los usuarios en entornos de realidad virtual. El diseño se compone de una plataforma (cinta mecánica) que se mueve en dos direcciones, adaptable al usuario. En el cual se le proporciona ante la cámara Kinect una serie de imagines reales para ofrecerle una experiencia inmersiva.

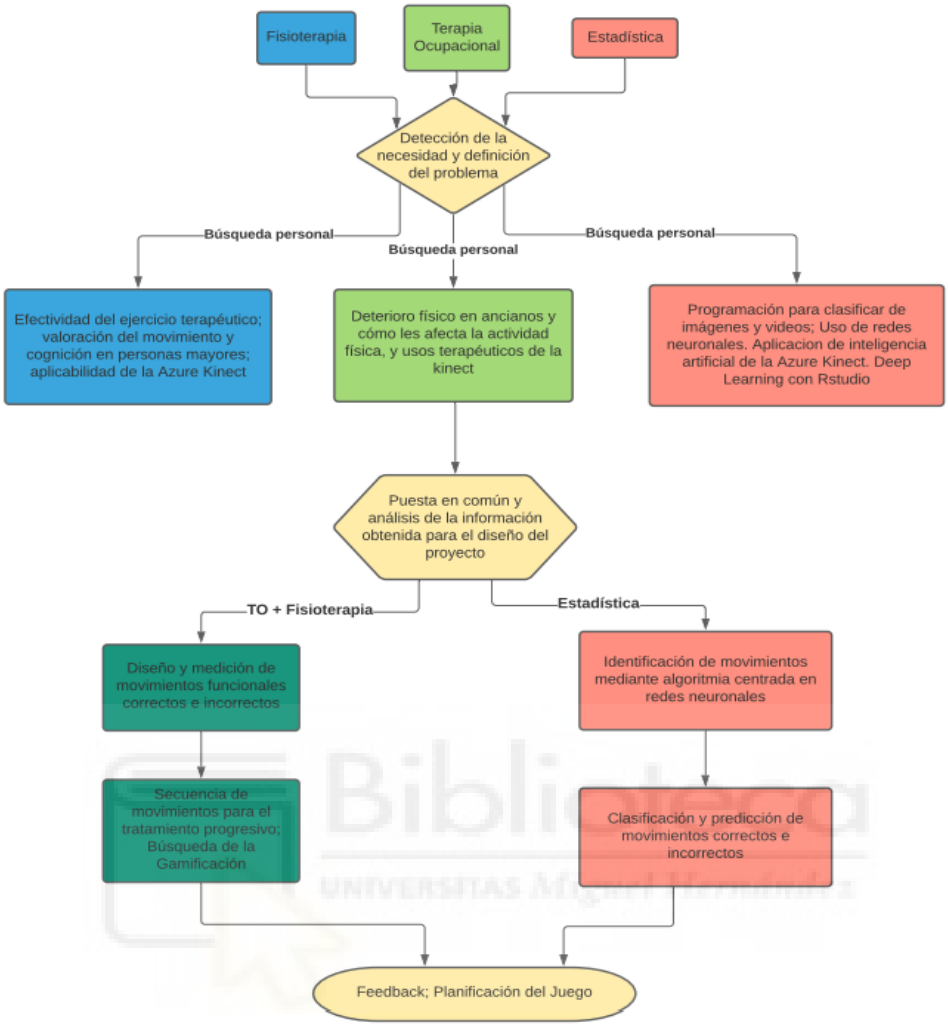


Figura 8: Esquema de la organización grupal

Fuente: Propia

En la figura anterior, vemos el propósito de los tres alumnos y sus labores personales en base a la idea principal. Después, se puede observar cómo pasan a una puesta en común y se va creando el proyecto hasta llegar a la planificación de los juegos y sus pautas.

Como final a este apartado, se puede ver como la idea principal se ha ido adaptando a los conocimientos de cada alumno y dando forma desde distintos ámbitos académicos, sin embargo, siempre complementándose de manera acertada ya que la idea no podria continuarse por separado. Las ideas salieron de juegos de Xbox, pero la diferenciación a otros proyectos es el enfoque de personas de tercera edad que por ciertos motivos han ido perdiendo su movilidad física y mental. Por lo tanto, los juegos no solo entrenan el cuerpo sino al mismo tiempo la mente.

Metodología específica:

En la parte bibliográfica la búsqueda se enfocó en la aplicación de la inteligencia artificial (IA) en sectores sanitarios, captación de personas en videos e imágenes, uso de redes neuronales en base a la programación de RStudio y el concepto de red neuronal de forma genérica para conocer el desarrollo y funcionamiento.

También se exploró buscadores académicos y de investigación para comprobar si había una idea similar para poder ver sus ventajas y hacer una comparación. Y en caso de tener debilidades tomarlo como ideas para mejorarlas en nuestro proyecto.

El concepto de IA en el sector de captación de imágenes tiene hoy en día amplios usos como, por ejemplo:

- Búsqueda de fotografías donde se realizan reconocimientos faciales.
- Gestión de casa inteligentes para capturar los movimientos dentro de la casa.
- Visión artificial en medicina (detección de tumores), seguridad (automatizar el control de acceso), control de calidad (sectores alimenticios), procesamiento de datos (reconocimiento de patrones), picking y packing (diferenciación de lotes).
- Video juegos

En referencia a las redes neuronales nos centramos en el libro "Deep Learning with R" de François Chollet. En este libro exploramos cuales son los conocimientos de la materia: el aprendizaje profundo, las redes neuronales y los fundamentos del aprendizaje automático.

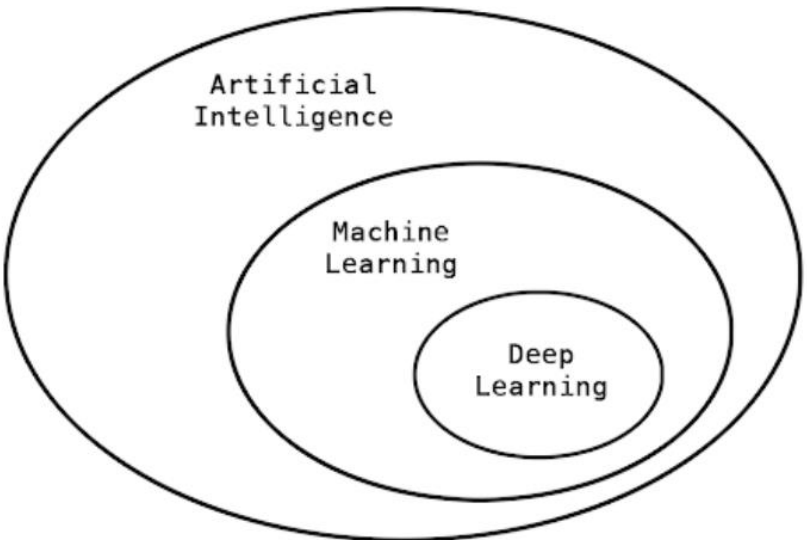


Figura 9: Imagen de los conocimientos que engloba la inteligencia artificial

Fuente: (Chollet & Allaire, 2017, pág. 2) [3]

Con esta imagen queremos evidenciar cual es la estructura de la inteligencia artificial. Ya que su consistencia es realizar tareas con la habilidad humana, pero esta vez con ordenadores. Así la medida de error es mínima, puesto que con los humanos es más complicado determinar qué tipo de error se tendrá.

La inteligencia artificial está compuesta por secuencia de ramas, una dentro de la otra. La primera es el aprendizaje automático y la segunda es el aprendizaje profundo. Reflexionando sobre ambos conceptos, iniciaremos sus explicaciones para entender mejor los argumentos.

- **Aprendizaje automático**: su aspecto principal es que los ordenadores que estudian los datos no están programados con anterioridad para extraer soluciones, es decir, van aprendiendo según el paso que van realizando al momento.
- **Aprendizaje profundo**: en este apartado se centra en el uso de las redes neuronales para reconocer y relacionar patrones complejos. Dichas redes usan capas para ir uniendo la información que se les cede. El tratamiento requiere de gran capacidad de entrenamiento y un gran conjunto de información.

Por otra parte, veremos qué es lo que se estudia en el aprendizaje automático, y cuál es su diferencia con la programación clásica a la que se estaba acostumbrado a realizar.

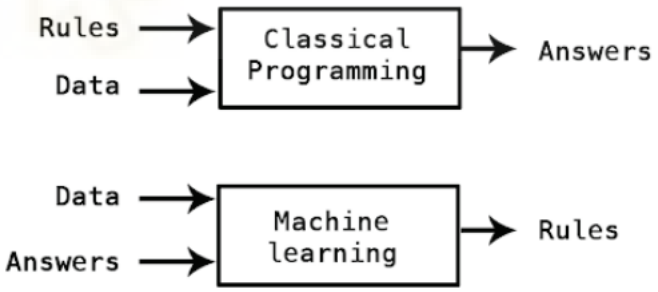


Figura 10: Estructura de la programación clásica con el aprendizaje automático

Fuente: (Chollet & Allaire, 2017, pág. 4) [3]

Según vemos, en la explicación del libro se observa cómo son dos tipos de procedimientos a seguir. El primero es un problema de programación clásica donde el usuario implanta sus reglas y datos mientras espera que le dé una solución. En cambio, con el aprendizaje automático sucede de otra manera. El humano ahora introduce las respuestas que quiere estudiar con sus datos y se espera a que le genere las reglas correspondientes. Estas, una vez obtenidas, se pretenden implantar a otros valores para que devuelvan la respuesta original.

Después para poner en marcha estas competencias, nos centramos en desenvolver la materia de las redes neuronales con clasificación binaria. Puesto que nuestro objetivo era pronosticar la clasificación de la actividad en dos casos: correctos (movimiento bueno) e incorrectos (movimiento malo). En el apartado de *Resultados* encontraremos las soluciones proporcionadas tras el entrenamiento de las redes neuronales y en el apartado *Anexos* se pondrá toda la algoritmia usada en Rstudio.

6. Información disponible

A continuación, se explicarán los variables que se usarán como lectura en el programa de RStudio. Tras generar los movimientos en la Kinect, la empresa proporcionó a la alumna los datos recopilados en diferentes documentos en formato de texto para la lectura. Las variables de estudio fueron las siguientes:

Frame	Puntos	Posición	Orientación	Confianza	Movimiento
Indica las partes fraccionadas a las que se reproduce el movimiento en la captación del video. Cada movimiento tiene diferentes frames (marco) según la duración del movimiento realizado.	Se refiere a cada una de las articulaciones del cuerpo. Los valores se comprenden entre el 0 al 31. Es decir, cada punto está representado en números. La tabla relacionada con los puntos de cada parte del cuerpo se encuentra en la parte de abajo.	Este apartado muestra las coordenadas X, Y, Z de cada una de las articulaciones que se detecta en la Kinect. Como los datos se pueden obtener en el rango de tiempo que se desee, entonces se obtiene el movimiento en el espacio de cada una de las articulaciones. La imagen indicada del espacio de cada punto se sitúa en la parte final de esta tabla.	Los datos son expresados en cuaterniones. Estos realmente señalan los ángulos de giro de las articulaciones. Es decir, nos proporciona la situación espacial y también los posibles ángulos de giro de cada movimiento.	Describe el grado de confianza que nos explica si los datos obtenidos de la cámara son medidas reales o si están estimadas por la inteligencia artificial. En este caso 1 significa que son datos estimados. Y el número 2 se refiere a que los datos son reales.	Será una variable binaria la cual constará de 0 cuando el movimiento sea correcto y 1 cuando sea incorrecto. Esto nos ayudará para la predicción del movimiento del paciente.

Figura 11: Estructura de la programación clásica con el aprendizaje automático

Fuente: Propia

Ahora se introducirán los valores con sus partes del cuerpo correspondientes que se relaciona con valores numéricos anteriores. En base a nuestros movimientos, el estudio de los datos se centró en las siguientes posiciones: 5, 6, 7, 8, 9, 11, 12, 13, 14, 15 y 16. Dichos nodos comprenden el rango de la cabeza a la cadera.

INDEX	JOINT NAME	PARENT JOINT
0	PELVIS	-
1	SPINE_NAVAL	PELVIS
2	SPINE_CHEST	SPINE_NAVAL
3	NECK	SPINE_CHEST
4	CLAVICLE_LEFT	SPINE_CHEST
5	SHOULDER_LEFT	CLAVICLE_LEFT
6	ELBOW_LEFT	SHOULDER_LEFT
INDEX	JOINT NAME	PARENT JOINT
7	WRIST_LEFT	ELBOW_LEFT
8	HAND_LEFT	WRIST_LEFT
9	HANDTIP_LEFT	HAND_LEFT
10	THUMB_LEFT	WRIST_LEFT
11	CLAVICLE_RIGHT	SPINE_CHEST
12	SHOULDER_RIGHT	CLAVICLE_RIGHT
13	ELBOW_RIGHT	SHOULDER_RIGHT
14	WRIST_RIGHT	ELBOW_RIGHT
15	HAND_RIGHT	WRIST_RIGHT
16	HANDTIP_RIGHT	HAND_RIGHT
17	THUMB_RIGHT	WRIST_RIGHT
18	HIP_LEFT	PELVIS
19	KNEE_LEFT	HIP_LEFT
20	ANKLE_LEFT	KNEE_LEFT
22	HIP_RIGHT	PELVIS
23	KNEE_RIGHT	HIP_RIGHT
24	ANKLE_RIGHT	KNEE_RIGHT
25	FOOT_RIGHT	ANKLE_RIGHT
26	HEAD	NECK
27	NOSE	HEAD
28	EYE_LEFT	HEAD
29	EAR_LEFT	HEAD
30	EYE_RIGHT	HEAD
31	EAR_RIGHT	HEAD

Figura 12: Tabla de valores de los nodos del cuerpo

Fuente: (Microsoft, 2021, págs. 30-31) [8]

Por otra parte, mostraremos los puntos anteriores en un dibujo para tener una idea más visual de los datos. Estos representan el cuerpo entero con los puntos detectables ante la cámara. Vemos que abarca variedad de puntos concretos como, por ejemplo: los ojos y las orejas y los pulgares de las manos. Aunque nuestra finalidad es realizar ejercicios que no conlleven captar movimientos tan finos para no arrastrar errores.

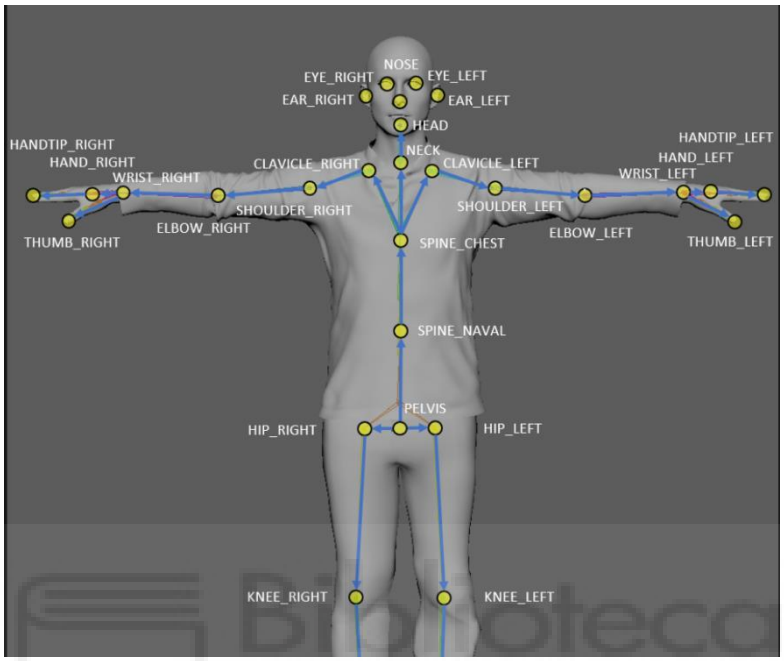


Figura 13: Imagen de los puntos del cuerpo en la Azure Kinect

Fuente: (Microsoft, 2021, pág. 30) [8]

Como indicamos antes, la localización de cada nodo se representará en coordenadas del espacio X, Y, Z y podemos ver como se refleja en esta imagen. Estas coordenadas nos ayudaran a captar los giros y tener una mejor precisión del movimiento.

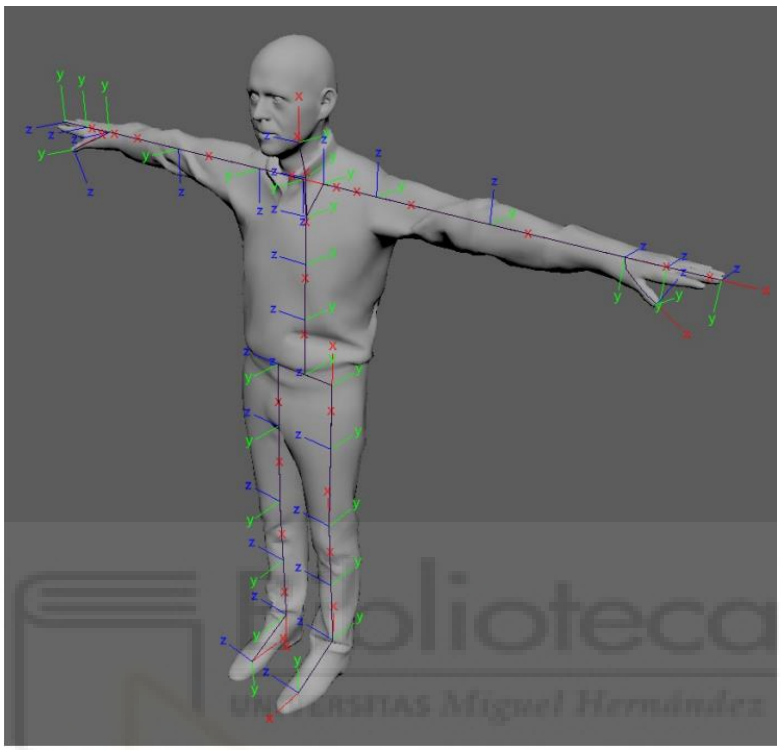


Figura 14: Imagen de los nodos con espacio X, Y, Z

Fuente: (Microsoft, 2021, pág. 30) [8]

Ya puesta en marcha la lectura de datos, el objetivo era realizar una matriz la cual incluya como valores horizontales todos los frames que se encuentran en un movimiento y verticalmente observar el resto de los valores como: puntos, orientación, movimiento…etc.

Sin embargo, para abarcar todos los movimientos que se realizaron y poder realizar la predicción se buscó agrupar todos los movimientos en un dataframe. En dicho dataframe se colocarán cada fichero en formato de texto como valor horizontal y los que se sitúan de manera vertical serán algunos valores anteriores, pero esta vez con la cantidad de frames correspondientes. Es decir, si el primer movimiento tenía 17 frames, pues lo valores de orientación X, Y, Z serian así X1, Y1, Z1, X2, Y2, Z3… X17, Y17 y Z17. Este mismo proceso lo realizaba el variable como orientación. El resto se mantenía en la forma original.

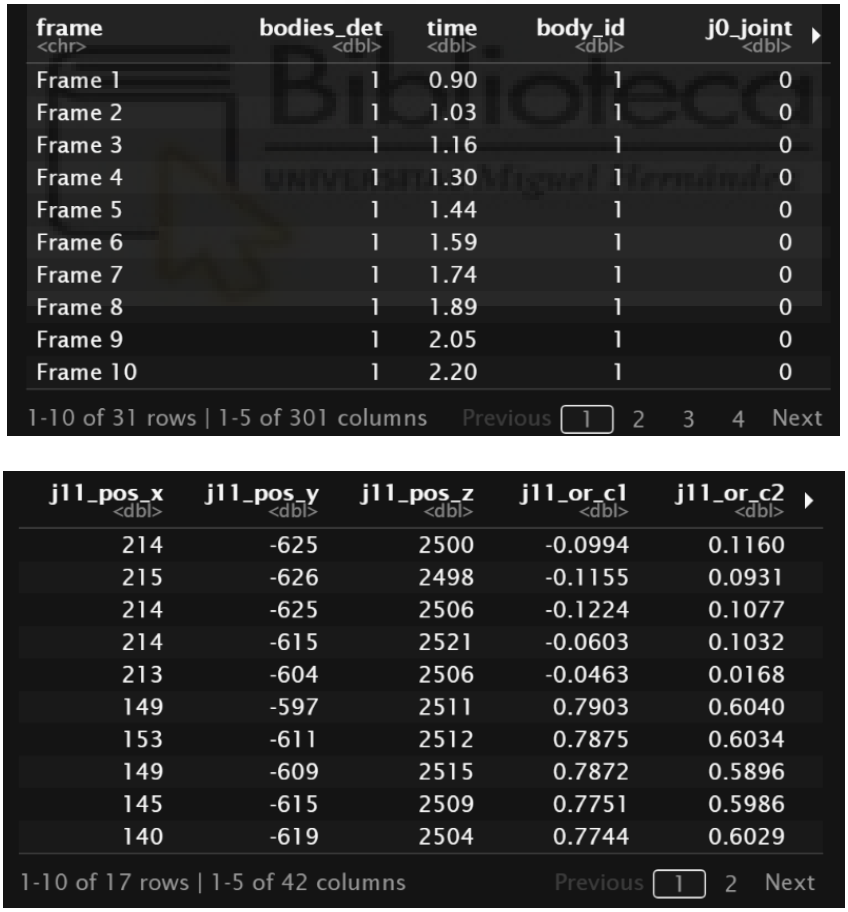
7. Resultados

7.1 Desarrollo del código

Iniciando este subapartado, comentaremos el código creado en el programa Rstudio y explicaremos cuales fueron las soluciones que nos salieron durante el proceso.

Comenzamos con el estudio de un único movimiento para poder ver el funcionamiento del algoritmo de una forma más sencilla. Después de probar la algoritmia, se procederá a realizar una lectura más rápida de los movimientos restantes. Se pretendía que el programa leyese tanto los movimientos clasificados como buenos y malos para poder luego unirlos en una misma tabla.

Principalmente, se creó una tabla con los valores de movimientos buenos realizados con alzamiento de brazo, anteriormente nombrados en la Figura 11. Dichos índices reflejarían unos pocos nodos, los que subrayamos en la Figura 12. Con esto se pretendía acotar los datos para centrarnos en estos y no arrastrar en un futuro posibles errores.



frame <chr>	bodies_det <dbl>	time <dbl>	body_id <dbl>	j0_joint <dbl>
Frame 1	1	0.90	1	0
Frame 2	1	1.03	1	0
Frame 3	1	1.16	1	0
Frame 4	1	1.30	1	0
Frame 5	1	1.44	1	0
Frame 6	1	1.59	1	0
Frame 7	1	1.74	1	0
Frame 8	1	1.89	1	0
Frame 9	1	2.05	1	0
Frame 10	1	2.20	1	0

1-10 of 31 rows | 1-5 of 301 columns Previous 1 2 3 4 Next

j11_pos_x <dbl>	j11_pos_y <dbl>	j11_pos_z <dbl>	j11_or_c1 <dbl>	j11_or_c2 <dbl>
214	-625	2500	-0.0994	0.1160
215	-626	2498	-0.1155	0.0931
214	-625	2506	-0.1224	0.1077
214	-615	2521	-0.0603	0.1032
213	-604	2506	-0.0463	0.0168
149	-597	2511	0.7903	0.6040
153	-611	2512	0.7875	0.6034
149	-609	2515	0.7872	0.5896
145	-615	2509	0.7751	0.5986
140	-619	2504	0.7744	0.6029

1-10 of 17 rows | 1-5 of 42 columns Previous 1 2 Next

Figura 15: Tablas creadas en Rstudio referente a los buenos movimientos

Fuente: Propia

Esta tabla expone los valores numéricos que se crean una vez captados por la Azure Kinect. Seleccionando primero los movimientos buenos.

Después, con la compilación que diseñamos, se procuró ajustar los valores sin generar registros nulos con la finalidad de obtener una misma dimensión de matrices tanto en valores buenos como malos movimientos.

Por tanto, cuando se forman las agrupaciones de datos y se trasponen los valores para iniciar una tabla, decidimos hacer lo mismo con los movimientos malos. Siguiendo el mismo proceso en el programa R, los resultados que nos muestra es una tabla parecida a la Figura 15, pero esta vez, con los valores de los desplazamientos malos.

Una vez realizado todo este proceso, pasaremos a la unión de ambas tablas con la finalidad de que se lean tanto las actividades de movimientos correctos como incorrectos. Concluida esta parte, pretendemos añadir otra columna de valor binario para indicar cuando el valor es 0 significa que el movimiento es correcto y cuando el valor es 1 es un movimiento incorrecto. Las cifras de las que hablamos se verán representadas en la siguiente figura.

j16_or_c4_p0.75 <dbl>	j16_or_c4_p1 <dbl>	movimiento <dbl>
0.845000	0.9263	0
0.819600	0.8513	0
0.739875	0.9172	0
0.822950	0.9322	0
0.905000	0.9281	0
0.481200	0.7738	0
0.724300	0.8704	0
0.784825	0.7982	0
0.206900	0.7866	0
0.641800	0.8576	0

31-40 of 40 rows | 210-212 of 211 c... Previous 1 2 3 4 Next

Figura 16: Tabla de ambos movimientos con columna de valores binarios

Fuente: Propia

Resultado:

Poniendo fin a la lectura inicial de datos en el subapartado anterior, podemos empezar con el uso de las redes neuronales. Centrándonos en la clasificación binara de “Chollet” (Chollet & Allaire, 2017) [3]. Este libro nos ayudó a generar la algoritmia adecuada y poder obtener los siguientes resultados.

Para aplicar las redes neuronales se necesita usar Tensorflow (Rosa, 2020) [10] en nuestro modelo, pero antes pondremos en contexto qué es y en que consiste su uso en nuestra algoritmia. Explicando su definición se dice que Tensorflow es un framework, es decir, un marco de trabajo en el cual nos enfrentaremos a problemas con ciertos criterios enfocados en situaciones específicas. Dicho conjunto de

trabajo nos facilitará la clasificación y reconocimiento de imágenes. Continuando con el concepto, nos adentramos a la descripción de la interfaz Keras que se pone en juego con Tensorflow que sirve para facilitarnos la aplicación de las herramientas que este posee. Estos dos atributos serán instalados previamente para proceder al entrenamiento de las redes neuronales.

Poniendo en juego la tabla creada anteriormente Figura 16, con los valores buenos y malos, extraeremos las matrices de entrenamiento y de testeo las cuales contendrán los siguientes datos.

- X_train: conjunto de 30 observaciones con 210 columnas.
- Y_train: conjunto de 10 observaciones con 210 columnas.
- X_test: conjunto de 30 observaciones recogiendo 1 columna con los datos binarios.
- Y_test: conjunto de 10 observaciones recogiendo 1 columna con los datos binarios.

Estos valores nos servirán como componentes de las capas iniciales, las cuales estas recogerán como valores de entrada y después transformándolos en las capas ocultas se extraerá unos valores de salida para explicar la clasificación de buenos o malos movimientos. Este procedimiento se define como modelo secuencial de Keras y aquí nosotros pondremos los valores de entrada y de las capas ocultas con el valor de activación “Sigmoid” que es el que se emplea cuando se hacen clasificaciones binarias.

A continuación, arrancaremos con la complicación del modelo secuencial indicando que tipo de función de pérdida, optimizador y métrica introduciremos. La primera indica como se medirá el rendimiento de las redes mientras estas están en su entrenamiento. La segunda explica como es el feedback de los valores durante el entrenamiento. Por último, la métrica, un valor en el cual nos centraremos durante el proceso que generalmente escoge la opción de “precisión”.

Para entrenar el modelo se usará la opción “Fit” donde indicaremos con la opción “Epoch” el número de iteraciones completas que se realizarán en el entrenamiento. “Batch_size” será la cantidad de datos que se usarán en cada iteración. Abajo se adjuntará el código de entrenamiento del modelo para después explicar las soluciones extraídas.

```
model <- keras_model_sequential() %>%
 layer_dense(units = 210, activation = "relu", input_shape =
ncol(x_train)) %>%
 layer_dense(units = 210, activation = "relu") %>%
 layer_dense(units = 1, activation = "sigmoid")
summary(model)

history <- model %>% compile(
  loss = 'binary_crossentropy',
  optimizer = 'adam',
  metrics = c('accuracy') )

history <- model %>% fit(
  x_train, y_train,
  epochs = 10, batch_size = 50,
  validation_split = 0.2
```

Figura 17: Código usando el modelo secuencial, la compilación y el ajuste

Fuente: Propia

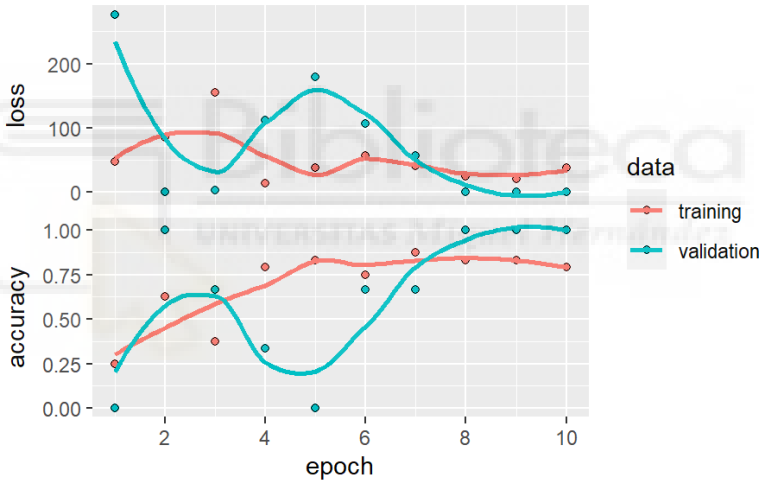


Figura 18: Gráfico de precisión de X_train e Y_train

Fuente: Propia

En la Figura 18 se observa como algunos puntos tanto en la gráfica de perdidas de red y de precisión están fuera de las líneas creadas durante las 10 iteraciones. Esto nos podria indicar que son los valores perdidos por no reconocer adecuadamente los movimientos durante el entrenamiento y su validación. Por lo que podemos indicar que esto sucede durante las iteraciones número 4 hasta la número 7.

Finalmente, tras entrenar varios modelos secuenciales este es el resultado más acertado que se ha tenido obteniendo una pérdida de red de 37.8201 y con una precisión de 0.791, es decir, del 79%.

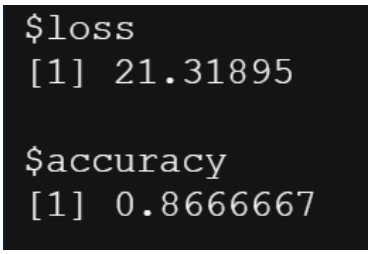


Figura 19: Resultado del conjunto de prueba

Fuente: Propia

Sin embargo, la precisión del conjunto de prueba que obtenemos a posteriori nos indica que es un poco más alta la presión de un 86%. Por lo que con una alta precisión podemos asegurar que el acierto en la lectura de movimientos correctos e incorrectos es moderadamente bueno.

8. Conclusión

8.1 Concepto de Gamificación

Ya culminando con el análisis, podemos decir que se pretende juntar todas las ideas en un solo concepto, es decir, la gamificación. Esto es un término que se entiende como métodos de enseñanza de forma interactiva mediante juegos / actividades. Consiste en motivar al usuario desarrollando sus capacidades cognitivas. Sus propiedades se basan en tener:

- **Un sistema de recompensas para incentivar al individuo**: basado en nuestra idea se buscó premiar de forma divertida teniendo en cuenta el tipo de grupo al que nos dirigimos que son las personas de la tercera edad.
- **Se genera un ranking para ver la habilidad de cada usuario**: para los usuarios más atrevidos que quieran comparar sus resultados con el resto de los participantes.
- **Asignar niveles para realizar las misiones**: contando con la edad, si hay patologías previas o alguna deficiencia se les asignará un tipo de actividad para no provocar un deterioro en su condición.
- **Puntuaciones para clasificar de manera cuantitativa**: al final de cada prueba el competidor obtendrá puntos si consigue dominar el juego adjudicado y este baremo podrá variar según avance los niveles.

Considerando dicha definición, se pensó en realizar juegos de rehabilitación porque se comprobó que son seguros y factibles para las personas mayores. Sin embargo, como se comentó en apartados anteriores se necesitan más documentación para verificar sobre la seguridad si se realizara en los hogares. Por lo que en un inicio se confeccionarían ante la supervisión de un médico.

8.2 Propuesta

Como conclusión a este proyecto, nuestro pensamiento es llevarlo a cabo mediante juegos programados por la compañera de Terapia Ocupacional, previamente evaluados por el compañero de Fisioterapia, y por último controlados y analizados por la alumna de Estadistica.

Los juegos contendrán contenido para ejercitar la mente, aunque la intención principal sea determinar la calidad del movimiento. El plan de juegos radica en diferentes niveles dependiendo de la flexibilidad del usuario.

El nombre de la aplicación sería "*Aging Rehab"* y el contenido de juegos terapéuticos son los siguientes:

Objetivo 1: **Colocar un plato en la estantería con una flexión de hombre de 150º.**

- Nivel principiante:
  El nombre para este minijuego es: *Einstein*.

Se inicia la postura con una flexión de hombro con hasta 90º con extensión completa de codo. Se trata de mover números que aparecerán en pantalla para poder clasificarlos de forma par e impar.

- Nivel intermedio:
  El nombre para este minijuego es: *Seguridad al volante*.
  Consiste en realizar el movimiento del hombro a 110º con extensión completa del codo, para ello se pensó un ejercicio de limpiar el parabrisas del coche con el brazo. En el caso de que realice dicho movimiento 3 veces de manera errónea el juego le lleva al paciente a realizar otra actividad de menor intensidad. La actividad que tendría que realizar seria golpear una pelota con flexión de 90º con extensión completa de hombro e ir aumentando gradualmente hasta conseguir realizar los grados que inicialmente se propusieron, es decir, 110º de flexión.

- Nivel avanzado:
  El nombre para este minijuego es: *Fernando Martín*.
  Aquí se debe encestar una pelota en una canasta. Al usuario se le contará como movimiento acertado si el hombro llega alcanzar los 130º de flexión hombro. Si no llegara a cumplir el propósito 3 veces seguidas, el propio juego le hará realizar otro ejemplo. Dicho ejercicio sería hacer una flexión de 110º con extensión completa de codo colocando relojes analógicos en la hora que se ponga en pantalla. Los relojes a medida que avanza la actividad se pondrán altos para que se llegue a completar los 130º de flexión.

- Nivel experto:
  El nombre para este minijuego es: *Salvador Dalí*.
  Se comienza con la sugerencia de pintar un cuadro consiguiendo la flexión de 150º de hombro. Como en los otros niveles, si el paciente falla 3 veces consecutivas se le adjudica otro juego. Este juego es parar balones que se lacen hacia el paciente así con el paso del juego llegue a realizar la flexión de hombro de 150º.
- Prueba final:
  El nombre para este minijuego es: *Estrella Michelin*.
  El paciente debe conseguir completar la actividad entera de coger el plato de la estantería con flexión de 150º de hombro y dejarlo en la mesa.

Objetivo 2: **Comer de forma autónoma.**

En dicho nivel se le propondrá cocinar una tortilla de patatas por pasos y cada uno le hará avanzar de nivel, teniendo que completar 4 niveles. El propósito final de esta actividad es llegar a completar la flexión y extensión del codo para evitar compensaciones futuras de manera que no se exija al paciente un movimiento intenso de primeras.

- Nivel 1: Se tendrá que pelar patatas y cortarlas para así tener en la mano no dominante la patata y con el brazo dominante hacer el movimiento de flexión y extensión del codo. Después su objetivo es meterlas al sartén.
  Como juego final para pasar al siguiente nivel se le hará jugar al típico juego de piedra, papel y tijera. Aquí deberá tener la mano no dominante en una línea que se pondrá en pantalla como apoyo. Y la dominante tendrá que formar un puño.

- Nivel 2: Se romperán los huevos para la tortilla haciendo la flexión de codo y manteniendo la flexión de hombro sobre los 90º. Después, se batirán haciendo flexión y extensión de codo con rotación interna de hombro.
  Para el siguiente nivel se tiene que hacer antes un ultimo ejercicio. Se iniciará con la separación de la baraja española por palos. Se debe realizar la extensión y flexión del codo con hombro de rotación interna.

- Nivel 3: Para continuar se debe mezclar los huevos y las patatas batiéndolos de nuevo. Por último, se debe verter la mezcla en la sartén.
  Para terminar este nivel se tiene que hacer esta actividad: Tocar el tambor con flexión de 80º de hombro y extensión de codo a una velocidad relativamente rápida.

- Nivel 4: Una vez cocinada la tortilla se debe darle la vuelta en el aire. Con este gesto conseguimos que se haga la flexión y extensión de hombro y codo a la vez.

Objetivo 3: **Peinarse haciendo el movimiento de 150º de hombro y 160º de codo.**

- Nivel principiante:
  Nombre de la actividad: *Juego de la rana.*
  La posición que debe hacer el participante es una flexión de codo y hombro de unos 90º. Para hacer esto tiene que tirar unas fichas para que se realice la extensión.

- Nivel medio:
  Nombre de la actividad: *Rafa Nadal.*
  Se inicia con la flexión de hombro hacia los 110º y una extensión completa de codo. Aquí como su propio nombre indica, deberá el usuario jugar al tenis llegando a hacer los 120º de flexión. Si este no consigue completar el ejercicio fallando 3 veces, se le derivará a otro ejercicio. Se le propondrá jugar al bingo, haciendo mover las fichas en diferentes posiciones de la pantalla para seleccionar los números indicados. Se parte de una flexión de hombro de 90º con extensión de codo que a medida que avance el juego se irá aumentando la flexibilidad de hombro de 110º y de 120º para el codo.

- Nivel avanzado:
  Nombre de la actividad: *Cámara y acción.*

En este nivel se tiene que llevar las manos a los ojos como si tuviese el usuario en sus manos una cámara. Con este movimiento se simulará que se realiza una foto en 3 segundos, así la actividad se completa con 90º de hombro y 140º de codo. Para poder volver a hacer la foto se debe hacer un gesto de 130º con el hombro y codo. La actividad le indicará al usuario que tiene que sacar 5 fotos, si este no cumple con el objetivo fallando 3 veces seguidas se le pone otra actividad diferente.
La actividad alternativa es dentro de una selva donde se encontrará por el camino ramas de árboles. Aquí se le propone que durante el trayecto el participante debe apartar las ramas para llegar a su destino final. Por lo que tiene que mover los brazos a diferentes lados con una flexión de 110º de hombro y 120º de codo. Se aumentará la dificultad para que se llegue a un máximo de 130º de flexión de hombre y 140º de flexión de codo.

- Nivel experto:
  Nombre de la actividad: *Rey del mar*.
  La posición que se desea conseguir es la de pescar 10 peces en una charca. El movimiento debe hacerse con una flexión de hombro 150º y de codo 160º. Como se inicia con las manos de detrás de la cabeza hacia la parte de delante se simulará el gesto de pescar. Si no consigue el reto se le pondrá otra actividad. Como propuesta alternativa se le pondrán en pantalla varios mosquitos volando sobre la cabeza del usuario y este tendrá que espantarlos. A la hora de ahuyentarlos con las manos se le guiará que la flexión tiene que ser de 150º para el hombro y 160º de codo.

- Prueba final:
  Nombre de la actividad: *Se tu propio peluquero.*
  Por último, se le adjudica al usuario la oportunidad de hacerse diferentes peinados para que los realice desde la frente hacia la nuca de manera autónoma.

Las puntuaciones para los juegos y niveles son las siguientes:

Con el nivel principiante se conseguirá 5 puntos, con el nivel medio se ganarán 10, con el avanzado 20, en el experto se adjudicarán 25 y para la prueba final si se completa se dan 40 puntos. Por lo tanto, se obtendrá un total de 100 puntos por cada objetivo, teniendo 3 objetivos que hacer. Cuando se finalizada cada nivel consiguiendo los puntos, nombrados anteriormente, en la pantalla aparecerá confeti y fuegos artificiales cuando se consigue el objetivo total.

También se pensó en ofrecer la selección de competencia por si algún usuario desee comparar sus resultados con la de los otros participantes. Por lo que, si los otros usuarios seleccionan esta opción se puedan ver en una tabla con su clasificación correspondiente.

Si la idea llegara a desarrollarse con compañía de la empresa Instead Technologies, se desearía implantar en centros de salud para que el chequeo de los pacientes sea dinámico y recreativo para la evaluación de la motricidad en personas de tercera edad.

El foco sería centros de salud donde se practiquen actividades dinámicas, de rehabilitación y puedan innovar introduciendo este mecanismo.

9. Bibliografía

[1]. Amadeus Albert, J., Owolabi, V., Gebel, A., Markus Brahms , C., Granacher, U., & Arnrich, B. (08 de 09 de 2020). *MDPI.* Obtenido de https://www.mdpi.com/1424-8220/20/18/5104

[2]. Campos Soberanis, M. (13 de 02 de 2018). *SoldAI.* Obtenido de https://medium.com/soldai/inspiraci%C3%B3n-biol%C3%B3gica-de-las-redes-neuronales-artificiales-9af7d7b906a

[3]. Chollet, F., & Allaire, J. (2017). *Deep Learning with R.* Manning Publications.

[4]. en:User:Cburnett. (07 de 05 de 2019). *Wikipedia.* Obtenido de Red neuronal artificial: https://es.wikipedia.org/wiki/Red_neuronal_artificial#/media/Archivo:Colored_neural_network_es.svg

[5]. Gómez Gil, M. (06 de 04 de 2016). *INAOE.* Obtenido de http://ccc.inaoep.mx/~pgomez/conferences/PggTSys16.pdf

[6]. Infasdev. (11 de 07 de 2018). *Infas Group*. Obtenido de https://www.infas.com.ar/repetibilidad-y-reproducibilidad/

[7]. Jorge Matich, D. (01 de 03 de 2011). Redes Neuronales. *Concepto Básicos y Aplicaciones*. Obtenido de https://www.frro.utn.edu.ar/repositorio/catedras/quimica/5_anio/orientadora1/monograias/matich-redesneuronales.pdf

[8]. Microsoft. (2021). *Azure Kinect DK .* Obtenido de https://opdhsblobprod01.blob.core.windows.net/contents/4a6d75bb3af747de838e6ccc97c5d978/008dc0f32af26971d918e371c651e690?skoid=2d004ef0-5468-4cd8-a5b7-14c04c6415bc&sktid=975f013f-7f24-47e8-a7d3-abc4752bf346&skt=2021-12-20T06%3A53%3A30Z&ske=2021-12-27T06%3

[9]. *Instead Technologies.* (s.f.). Obtenido de http://www.instead-technologies.com/products/nemty/

[10]. Rosa, J. (29 de 03 de 2020). *Tensorflow y Keras con R.* Obtenido de https://www.johan-rosa.com/post/tensorflow-y-keras-con-r/

10. Anexos

En este apartado se expondrá todo el código que se empleó en el software Rstudio para el uso de redes neuronales. Actualmente, se está desarrollando toda la recopilación de datos para hallar una solución en la predicción de movimientos.

Librerías usadas

```
library(dplyr)
library(readr)
library(tidyr)
library(tidyverse)
library(tensorflow)
library(keras)
library(caret)
```

Directorios

```
dirbase="C:/Users/lili_/OneDrive/Escritorio/TFG/datos"
#Directorio de localización de los datos
dir_brazoalto_buenos=paste0(dirbase,"/brazo_alto/buenos_movimientos/")
dir_brazoalto_malos=paste0(dirbase,"/brazo_alto/malos_movimientos/")
```

Seleccionamos buenos movimientos

```
mydir=dir_brazoalto_buenos
#Leemos los nombres de ficheros .txt en el directorio
file.list <- list.files(path=mydir,pattern='*.txt')
n<-length(file.list);n #es el número de ficheros disponibles

#Generamos una lista y cargamos en cada elemento el contenido de un fichero a modo de vector "chr"
file.list=sapply(file.list,function(x) paste0(mydir,x))
df.list <- lapply(file.list, read_lines)
```

#No todos los ficheros tienen el mismo número de frames (ligado a tiempo), y podría no tener el mismo número de nodos (joint)

#funcion para contar el número de Frames en cada fichero

```
n_frames=function(x){
  sum(str_count(x,"Frame"))}
```

#funcion para contar número de Joint en cada fichero

```
n_joints=function(x){
  sum(str_count(x,"Joint"))/sum(str_count(x,"Frame"))}
```

```
nframes=unlist(lapply(df.list,n_frames))
njoints=unique(unlist(lapply(df.list,n_joints)))
```

#Cada elemento de la lista lo transformó en una matriz con tantas filas como frames y tantas columnas como datos disponibles para cada frame

#Las columnas son: frame,bodies_det, time,body_id,joint,y las 8 medidas numéricas en cada nodo (joint): Position(xyz), Orientation(c1c2c3c4), Confidence

#Para ello, basta aplicar, a cada elemento de la lista, la siguiente función, que transforma un vector con todos los datos leídos en un fichero, en la mencionada matriz.

```
datos=df.list[[1]]
read_video_kinect=function(datos){
  # datos es un vector con todo el contenido de un fichero de vídeo, tal como es leído por read_lines

  # Calculamos el número de frames y de joints por frame
  nframe=sum(str_count(datos,"Frame"))
  njoint=sum(str_count(datos,"Joint"))/nframe
```

```
# Eliminamos todos los caracteres no deseados
  datos=datos %>%
  str_replace_all(":",";") %>%
  str_replace_all(",",";") %>%
  #str_replace_all(";","") %>%
  str_replace_all("Joint","") %>%
  str_replace_all("Position","")  %>%
  str_replace_all("Orientation","")  %>%
  str_replace_all("Confidence","")  %>%
  str_replace_all("mm","")  %>%
  str_replace_all("\\[","")  %>%
  str_replace_all("\\]","")  %>%
  str_replace_all("\\(","")  %>%
  str_replace_all("\\)","")   %>%
  str_replace_all("--","")  %>%
  str_replace_all("bodies detected","")  %>%
  str_replace_all("Body ID","")  %>%
  str_replace_all("Time","")


# identificamos las posiciones en las que se inicia y se termina cada Frame
ini_frame=str_which(datos,"Frame")
# 4 es el número extra de identificadores en cada frame: frame, bodies detected, body id, time
fin_frame=ini_frame+(4+njoint)-1


sel=ini_frame[1]:fin_frame[1]
for(i in 2:nframe){
  sel=c(sel,ini_frame[i]:fin_frame[i])}


# Para nombrar sus columnas creamos los nombres de los nodos y extras para cada frame
```

```
nodos_names=paste0("j",0:(njoint-1),"_")
names_var=c("frame","bodies_det","time","body_id",nodos_names)
# Definimos una matriz para el vídeo, en la que cargamos, por filas, la info de cada
frame
mat_video=t(array(datos[sel],dim=c(length(names_var),nframe)))
colnames(mat_video)=names_var

# la transformamos en tibble para manipularlo mejor
mat_video=as_tibble(mat_video)

# nombres de las columnas que descomponemos para cada joint
joint_var=c("joint","pos_x","pos_y","pos_z","or_c1","or_c2","or_c3","or_c4","conf")

#Solamente para las columnas que contienen la información de los nodos,
#separamos los datos relativos a cada nodo descomponiendo en columnas
#y borramos la columna original
for(j in nodos_names){

mat_video=separate(mat_video,col=!!j,into=paste0(j,joint_var),sep=";",remove=TRUE)
}

# eliminamos carácteres no deseados
mat_video=mat_video %>%
  mutate(time=str_replace(time,";",""),
         body_id=str_replace(body_id,";",""))

#transformamos a numéricos todos los valores salvo la etiqueta del frame en la columna
1
mat_video[,-1]=apply(mat_video[,-1],2,FUN=as.numeric)
return(as_tibble(mat_video))}
read_video_kinect(df.list[[1]])
```

Lectura de la información de los ficheros

```
#Creo una lista en la que cada elemento es una matriz con toda la información
procesada de cada vídeo, en formato de matriz: filas=frames, columnas=info del frame
video.list <- lapply(df.list, read_video_kinect)
tail(video.list)

#cuidado porque vido.list es una lista y no se le puede aplicar el pipe %>%
#Hay que manipularla con lapply o sapply
```

Identificación de nodos

```
#No todos los nodos son activos para un determinado movimiento.
#Hay que identificar cuáles son los nodos activos, para concentrarse en ellos y
simplificar así el problema de comparación.
vars=c("_pos_x","_pos_y","_pos_z","_or_c1","_or_c2","_or_c3","_or_c4")
nodos_activos= paste0(rep(paste0("j",11:16),rep(7,6)),vars)

seleccionar_nodos=function(datosm){
  datosm=as_tibble(datosm)
  datosm=datosm %>%
    select(nodos_activos)
  return(datosm)
}

#seleccionamos los nodos activos y sus variables en todos los videos (en todos los
elementos de la lista)
video.list.sel=lapply(video.list,FUN=seleccionar_nodos)
video.list.sel[[1]]

####DIMENSIONAMIENTO SIMILAR PARA TODOS LOS VIDEOS####
#Al tener diferentes frames en cada video, interesaría resumir cada nodo con
percentiles, con el fin de tener en todos los videos, idéntica dimensión
```

```
resumir_per=function(datosm,perc=c(0,0.25,0.5,0.75,1)){
 apply(datosm,2,quantile,probs=perc)
}

# Construimos un único array con tantas filas como ficheros y con tantas columnas como información en los nodos
# seleccionados, resumida en los 5 percentiles calculados
video.array=t(sapply(video.list.sel,resumir_per))
percentiles=paste0("_p",c(0,0.25,0.5,0.75,1))
varnames=paste0(rep(nodos_activos,rep(length(percentiles),length(nodos_activos))),percentiles)
colnames(video.array)=varnames
```{r , resultadosbuenos}
video.array.buenos=video.array
```

Seleccionamos malos movimientos

```
mydir=dir_brazoalto_malos
#Leemos los nombres de ficheros .txt en el directorio
file.list <- list.files(path=mydir,pattern='*.txt')
n<-length(file.list);n #es el número de ficheros disponibles

#Generamos una lista y cargamos en cada elemento el contenido de un fichero a modo de vector "chr"
file.list=sapply(file.list,function(x) paste0(mydir,x))
df.list <- lapply(file.list, read_lines)

#No todos los ficheros tienen el mismo número de frames (ligado a tiempo), y podría no tener el mismo número de nodos (joint) funcion para contar el número de Frames en cada fichero
n_frames=function(x){
  sum(str_count(x,"Frame"))}
```
```

```
#funcion para contar número de Joint en cada fichero
n_joints=function(x){
  sum(str_count(x,"Joint"))/sum(str_count(x,"Frame"))}

nframes=unlist(lapply(df.list,n_frames))
njoints=unique(unlist(lapply(df.list,n_joints)))
```

#Cada elemento de la lista lo transformo en una matriz con tantas filas como frames y tantas columnas como datos disponibles para cada frame

#Las columnas son: frame,bodies_det, time,body_id,joint, y las 8 medidas numericas en cada nodo (joint): Position(xyz), Orientation(c1c2c3c4), Confidence

#Para ello, basta aplicar, a cada elemento de la lista, la siguiente función, que transforma un vector con todos los datos leídos en un fichero, en la mencionada matriz.

```
datos=df.list[[1]]
read_video_kinect=function(datos){
  # datos es un vector con todo el contenido de un fichero de vídeo,
  # tal como es leído por read_lines
  # Calculamos el número de frames y de joints por frame
  nframe=sum(str_count(datos,"Frame"))
  njoint=sum(str_count(datos,"Joint"))/nframe

  # Eliminamos todos los caracteres no deseados
  datos=datos %>%
  str_replace_all(":",";") %>%
  str_replace_all(",",";") %>%
  #str_replace_all(";","") %>%
  str_replace_all("Joint","") %>%
  str_replace_all("Position","")  %>%
  str_replace_all("Orientation","")  %>%
  str_replace_all("Confidence","")  %>%
  str_replace_all("mm","")  %>%
```

```
  str_replace_all("\\[","")  %>%
  str_replace_all("\\]","")  %>%
  str_replace_all("\\(","")  %>%
  str_replace_all("\\)","")   %>%
  str_replace_all("--","")  %>%
  str_replace_all("bodies detected","")  %>%
  str_replace_all("Body ID","")  %>%
  str_replace_all("Time","")


# identificamos las posiciones en las que se inicia y se termina cada Frame
ini_frame=str_which(datos,"Frame")
# 4 es el número extra de identificadores en cada frame: frame, bodies detected, body id, time
fin_frame=ini_frame+(4+njoint)-1


sel=ini_frame[1]:fin_frame[1]
for(i in 2:nframe){
  sel=c(sel,ini_frame[i]:fin_frame[i])}


# Para nombrar sus columnas creamos los nombres de los nodos y extras para cada frame
nodos_names=paste0("j",0:(njoint-1),"_")
names_var=c("frame","bodies_det","time","body_id",nodos_names)
# Definimos una matriz para el vídeo, en la que cargamos, por filas, la info de cada frame
mat_video=t(array(datos[sel],dim=c(length(names_var),nframe)))
colnames(mat_video)=names_var


# la transformamos en tibble para manipularlo mejor
mat_video=as_tibble(mat_video)
```

```
# nombres de las columnas que descomponemos para cada joint
joint_var=c("joint","pos_x","pos_y","pos_z","or_c1","or_c2","or_c3","or_c4","conf")


#Solamente para las columnas que contienen la info de los nodos, separamos los datos relativos a cada nodo descomponiendo en columnas y borramos la columna original
for(j in nodos_names){

mat_video=separate(mat_video,col=!!j,into=paste0(j,joint_var),sep=";",remove=TRUE)
}
# eliminamos caracteres no deseados
mat_video=mat_video %>%
  mutate(time=str_replace(time,";",""),
         body_id=str_replace(body_id,";",""))


#transformamos a numéricos todos los valores salvo la etiqueta del frame en la columna 1
mat_video[,-1]=apply(mat_video[,-1],2,FUN=as.numeric)


return(as_tibble(mat_video))}
read_video_kinect(df.list[[1]])


####LECTURA DE LA INFORMACIÓN DE TODOS LOS FICHEROS####
#Creo una lista en la que cada elemento es una matriz con toda la informacion
procesada de cada vídeo, en formato de matriz: filas=frames, columnas=info del frame
video.list <- lapply(df.list, read_video_kinect)
tail(video.list)


#cuidado porque video.list es una lista y no se le puede aplicar el pipe %>% Hay que manipularla con lapply o sapply


####IDENTIFICACION DE NODOS ACTIVOS###
#No todos los nodos son activos para un determinado movimiento.
```

```
#Hay que identificar cuáles son los nodos activos, para concentrarse en ellos y simplificar así el problema de comparación.

vars=c("_pos_x","_pos_y","_pos_z","_or_c1","_or_c2","_or_c3","_or_c4")

nodos_activos= paste0(rep(paste0("j",11:16),rep(7,6)),vars)


seleccionar_nodos=function(datosm){
  datosm=as_tibble(datosm)
  datosm=datosm %>%
    select(nodos_activos)
  return(datosm)
}


#seleccionamos los nodos activos y sus variables en todos los videos (en todos los elementos de la lista)

video.list.sel=lapply(video.list,FUN=seleccionar_nodos)

video.list.sel[[1]]


####DIMENSIONAMIENTO SIMILAR PARA TODOS LOS VIDEOS####

#Al tener diferentes frames en cada video, interesaria resumir cada nodo con percentiles, con el fin de tener en todos los videos, identica dimensión


resumir_per=function(datosm,perc=c(0,0.25,0.5,0.75,1)){
 apply(datosm,2,quantile,probs=perc)
}

# Construimos un único array con tantas filas como ficheros y con tantas columnas como información en los nodos seleccionados, resumida en los 5 percentiles calculados

video.array=t(sapply(video.list.sel,resumir_per))

percentiles=paste0("_p",c(0,0.25,0.5,0.75,1))

varnames=paste0(rep(nodos_activos,rep(length(percentiles),length(nodos_activos))),percentiles)

colnames(video.array)=varnames
```

```{r , resultadosmalos}
video.array.malos=video.array
```

Seleccionamos buenos y malos movimiento y lo convertimos en data Frame

```
video.df.buenos=data.frame(video.array.buenos)
video.df.malos=data.frame(video.array.malos)

# verificamos que tienen las mismas columnas
colnames(video.df.malos)==colnames(video.df.buenos)

# pegamos unos debajo de otros
video.df = rbind(video.df.buenos,video.df.malos)
# y añadimos una última columna para identificar los buenos (1) de los malos
video.df$movimiento = rep(c(1,0), c(nrow(video.df.buenos),nrow(video.df.malos)))
```

Entrenamos las redes neuronales y ajustamos el modelo

```
#Utiliza la algoritmia que sale en Chollet para clasificacion binaria.
index <- createDataPartition(video.df$movimiento, p=0.75, list=FALSE)
final.training <- video.df[index,]
final.test <- video.df[-index,]

x_train = final.training[,- ncol(final.training)]
y_train = final.training$movimiento
x_test = final.test[,- ncol(final.test)]
y_test = final.test$movimiento
x_train = as.matrix(x_train)
y_train = as.matrix(y_train)
str(x_train)
```

```
model <- keras_model_sequential() %>%
 layer_dense(units = 210, activation = "relu", input_shape = ncol(x_train)) %>%
 layer_dense(units = 210, activation = "relu") %>%
 layer_dense(units = 1, activation = "sigmoid")
summary(model)

history <- model %>% compile(
  loss = 'binary_crossentropy',
  optimizer = 'adam',
  metrics = c('accuracy') )

history <- model %>% fit(
  x_train, y_train,
  epochs = 10, batch_size = 50,
  validation_split = 0.2
)
plot(history)
model %>% evaluate(x = x_train, y = y_train, verbose = 0)
```