Universidad de Alcalá
Escuela Politécnica Superior

**Grado en Ingeniería Electrónica de Comunicaciones**

**Trabajo Fin de Grado**

# Detección de acciones humanas a partir de información de profundidad mediante redes neuronales convolucionales



**Autor:** Sergio de López Diz

**Tutores:** Cristina Losada Gutiérrez y David Fuentes Jiménez

2019

# UNIVERSIDAD DE ALCALÁ

ESCUELA POLITÉCNICA SUPERIOR

**Grado en Ingeniería Electrónica de Comunicaciones**

**Trabajo Fin de Grado**

**Detección de acciones humanas a partir de información de profundidad mediante redes neuronales convolucionales**

Autor: Sergio de López Diz

Tutores: Cristina Losada Gutiérrez y David Fuentes Jiménez

**Tribunal:**

**Presidente:** Daniel Pizarro Pérez

**Vocal 1º:** Alfredo Gardel Vicente

**Vocal 2º:** Cristina Losada Gutiérrez

Calificación: .....

Fecha: .....

*"Uno de los grandes descubrimientos que un hombre puede hacer, una de sus grandes sorpresas, es encontrar que puede hacer lo que temía que no podía hacer."*

Henry Ford

# Agradecimientos

*Mi padre me explicó que la educación y el conocimiento*
*es lo que le permitirá a los niños mejorar el mundo.*

Steve Wozniak

En primer lugar, me gustaría agradecer a mi familia y mi novia toda la comprensión, apoyo y ayuda que me han proporcionado durante estos 4 años de carrera. Os quiero Mamá, Papá, Alba y María.

Por otro lado, quiero agradecer a mi tutora Cristina, co-tutor David y mi fiel amigo Robert todo el esfuerzo en el desarrollo de este trabajo, así como a los profesores y amigos del laboratorio. Gracias a todos.

# Resumen

El objetivo principal del presente trabajo es la implementación de un sistema de detección de acciones humanas en el ámbito de la seguridad y la video-vigilancia a partir de la información de profundidad (*"Depth"*) proporcionada por sensores RGB-D. El sistema se basa en el empleo de redes neuronales convolucionales 3D (3D-CNN) que permiten realizar de forma automática la extracción de características y clasificación de acciones a partir de la información espacial y temporal de las secuencias de profundidad. La propuesta se ha evaluado de forma exhaustiva, obteniendo como resultados experimentales, una precisión del 94 % en la detección de acciones.

Si tenéis problemas, sugerencias o comentarios sobre el mismo, dirigidlas por favor a Sergio de López Diz <s.lopezd@edu.uah.es>.

**Palabras clave:** Profundidad, Acciones, 3D-CNN.

# Abstract

The main objective of this work is the implementation of human actions detection system in the field of security and video-surveillance from depth information provided by RGB-D sensors. The system is based on 3D convolutional neural networks (3D-CNN) that allow the automatic features extraction and actions classification from spatial and temporal information of depth sequences. The proposal has been exhaustively evaluated, obtaining as experimental results, an accuracy of 94 % in the actions detection.

If you have problems, suggestions or comments on the document, please forward them to Sergio de López Diz <s.lopezd@edu.uah.es>.

**Keywords:** Depth, Actions, 3D-CNN.

# Índice general

# Índice de figuras

# Índice de tablas

# Índice de algoritmos

# Lista de acrónimos

| | |
|---|---|
| 3D-CNN | redes neuronales convolucionales 3D. |
| CCE | Categorical Cross Entropy. |
| CNN | Red Neuronal Convolucional. |
| CNNs | Redes Neuronales Convolucionales. |
| DNN | Redes Neuronales Profundas. |
| GPU | Graphics Processing Unit. |
| LOPD | Ley Orgánica de Protección de Datos de Carácter Personal. |
| LSTM | Long Short Term Memory. |
| MAE | Error medio absoluto. |
| MSE | Error medio cuadrático. |
| NNs | Redes Neuronales. |
| RGB | Red-Green-Blue. |
| RGB-D | Red-Green-Blue-Depth. |
| RNN | Redes Neuronales Recurrentes. |
| SGD | Stochatic Gradient Descent. |
| TFG | Trabajo Fin de Grado. |
| ToF | Time of Flight. |

# Capítulo 1

# Introducción

*La paciencia es un elemento clave del éxito*

Bill Gates

## 1.1 Introducción

En el contexto de la visión artificial, el reconocimiento de acciones ha cobrado una gran importancia en los últimos años, debido principalmente, a sus múltiples aplicaciones en el estudio del comportamiento humano entre las que se incluyen las referidas a la vida asistida en hogares inteligentes (*smart homes*), monitorización sanitaria, deportes, seguridad y video-vigilancia, por lo que ha atraído la atención de muchos investigadores [15–17]. Gran parte de los trabajos se basan en uso de cámaras de color [18, 19], sin embargo, recientemente han cobrado importancia los sensores Red-Green-Blue-Depth (RGB-D) que, además de una imagen de color, proporcionan un mapa de profundidad [20, 21], este hecho ha dado lugar a un gran número de trabajos que emplean esta tecnología para el reconocimiento de acciones [22–24].

Las diferentes propuestas comentadas proporcionan buenos resultados en condiciones controladas, sin embargo, presentan problemas en escenarios con un alto grado de oclusiones. Además, el hecho de emplear cámaras de color RGB o RGB-D implica la existencia de información que permite la identificación de los usuarios, por lo que pueden aparecer problemas relacionados con la privacidad debido a las leyes vigentes. Por otro lado, los sensores de profundidad [21, 25] permiten obtener información de distancia de cada punto de la escena a la cámara mediante la medida indirecta del tiempo de vuelo de una señal infrarroja modulada. El uso de este tipo de sensores permite preservar la privacidad de las personas, al no ser posible reconocer su identidad con la información proporcionada. En la figura 1.1 se muestra una imagen de profundidad a través de un mapa de color que permite observar de forma más clara las diferencias de contraste, las cuáles son insuficientes para el reconocimiento de la persona.

Por otra parte, cabe destacar que gracias a la mejora experimentada en la tecnología de procesamiento de datos en los últimos años, se han desarrollado con mucha fuerza los trabajos basados en Redes Neuronales Profundas (DNN), especialmente para aplicaciones de clasificación [26] y regresión [27]. En el caso del reconocimiento de acciones, se emplean frecuentemente Redes Neuronales Recurrentes (RNN) que incluyen la componente temporal para la extracción de características, tanto con arquitecturas Long Short Term Memory (LSTM) [28, 29] como basadas en capas convolucionales 3D [30, 31] utilizando la información de color RGB y la de profundidad (*Depth*). Los trabajos basados en *Deep Learning* extraen características comenzando por un nivel de abstracción bajo y aumentando la misma a medida que se

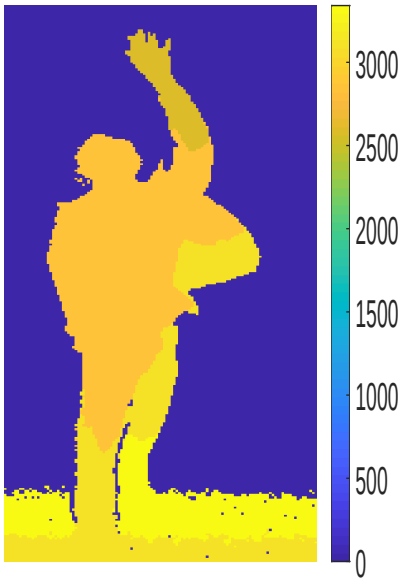



Figura 1.1: Ejemplo de imagen de profundidad que permite eludir problemas relacionados con la privacidad de las personas.

avanza por las capas. Este tipo de sistemas disponen de dos fases bien diferenciadas, la fase de extracción de características en la cual mediante el uso principalmente de filtros convolucionales entrenados y capas de filtrado se encargan de aprender y extraer las principales características concernientes al problema de interés, para posteriormente pasar a la fase de predicción en la cual se utilizan dichas características para clasificar o predecir una salida.

En este contexto, el objetivo del presente Trabajo Fin de Grado (TFG) es la detección de acciones, realizando tanto la extracción de características como la clasificación en un único bloque caracterizado por una 3D-CNN, utilizando únicamente la información de profundidad proporcionada por un sensor basado en tiempo de vuelo (ToF) [21, 25]. A lo largo del presente documento, se describe la implementación y evaluación de un sistema de detección de acciones cuya entrada son vídeos con información de profundidad, que se procesan e introducen en una CNN obteniendo como salida la clasificación de la acción entrante al sistema.

Se debe mencionar la dificultad que se presenta en la definición de las acciones, pues cada una de ellas tiene un tiempo de ejecución distinto y puede existir solapamiento entre ellas. Es por ello que el tiempo escogido para detectar una determinada acción es crucial pues si se trabaja con tiempos bajos, puede suceder que la acción no se muestre completa por las diferencias en el tiempo de ejecución de las acciones consideradas. En contraposición, si se emplean tiempos de análisis de acción demasiado elevados pueden aparecer solapamiento entre ellas.

Tanto el entrenamiento, como la validación y el test, se han obtenido con la base de datos *NTU RGB+D Action Recognition Dataset* [13,14], puesta a disposición de la comunidad científica por el ROSE Lab de la Nanyang Technological University de Singapore. Se ha elegido esta base de datos debido a que proporciona un gran número de vídeos, tanto con información RGB como de profundidad, que incluyen numerosas personas realizando diferentes acciones, lo que permite entrenar y validar el sistema propuesto en este trabajo. Además, se han realizado pruebas experimentales con la base de datos *UTKinect-Action3D* [32] con el objetivo de validar el sistema frente a cambios en el punto de vista y escenario.

## 1.2 Sistema propuesto

En este apartado se presenta el sistema propuesto de detección de acciones incluyendo las fases necesarias para su implementación. El diagrama de trabajo propuesto se muestra en la figura 1.2.

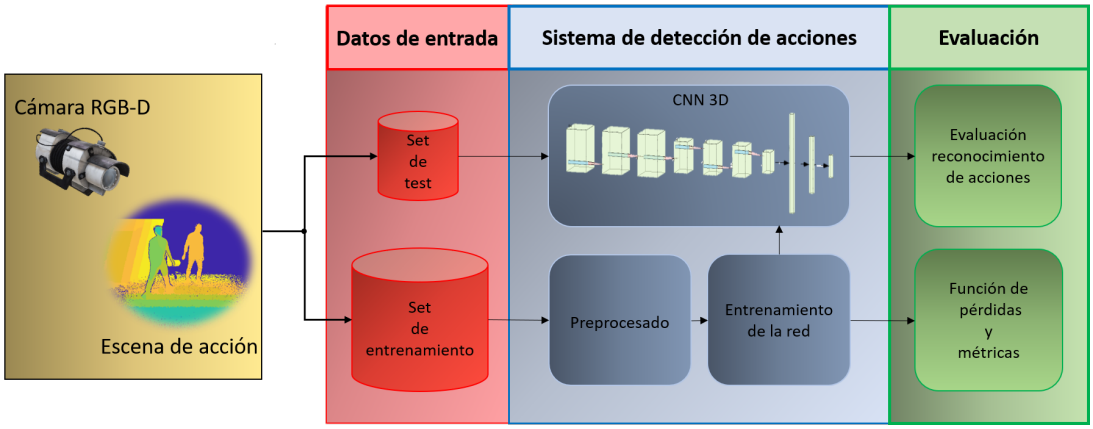



Figura 1.2: Esquema general del sistema propuesto para el reconocimiento de acciones en secuencias de imágenes de profundidad.

Cada una de las etapas que forman el sistema reflejado en la figura 1.2 se describen brevemente a continuación, y se explican con mayor detalle en el capítulo 4.

1. **Preparación de los datos de entrada a la red:** En esta fase se realiza la creación de la base de datos tanto para entrenamiento como para la validación del funcionamiento del sistema.

   (a) **Elección de las acciones a detectar:** se trata de un punto clave para la construcción del núcleo del sistema de detección de acciones, pues define el número de neuronas de la capa de salida. Como se ha comentado en el apartado anterior, el sistema se centra en el reconocimiento de acciones en el contexto de la seguridad. En la tabla 1.1 se muestra un resumen de las acciones seleccionadas de la base de datos NTU relativas a la video-vigilancia con su correspondiente código asignado. Se han desechado aquellas relativas a un contexto de acciones cotidianas.

| **Acciones utilizadas de la base de datos NTU** | |
|---|---|
| **Acción** | **Código** |
| Propinar un puñetazo | A01 |
| Levantarse | A02 |
| Caerse | A03 |
| Sentarse | A04 |
| Separarse de una persona | A05 |
| Propinar una patada | A06 |
| Empujar | A07 |
| Andar hacia una persona | A08 |
| Lanzar un objeto | A09 |

Tabla 1.1: Resumen de aciones a detectar por el sistema propuesto.

   (b) **División de la base de datos:** se debe decidir que número de secuencias se emplean para realizar el entrenamiento de la red neuronal y cuáles para evaluar el funcionamiento del sistema. En el capítulo de 4 se expone en detalle la elección de la división de la base de datos empleada.

   (c) **Preprocesado:** es necesaria la aplicación de distintas operaciones de preprocesado en las imágenes de entrada para proporcionar a la red una secuencia adecuada entre las que se

incluyen la normalización y el escalado de valores con el objetivo de evitar posibles saturaciones en el desempeño del sistema.

2. **Entrenamiento de la red neuronal:** es la etapa fundamental para la consecución de un sistema robusto y con un funcionamiento correcto en la tarea de la detección de acciones. Se deben definir un conjunto de parámetros que permitan el correcto entrenamiento de la red, como son el tamaño de *batch* y el número de épocas, además de determinar la función de pérdidas a emplear.

3. **Evaluación del sistema:** la evaluación del sistema consiste en la obtención de resultados que permitan determinar el comportamiento de la red a lo largo de todas las épocas de entrenamiento y sacar el valor de precisión obtenida en la clasificación de las distintas acciones. Mediante el set de test creado, se generan unas métricas de resultados con respecto al comportamiento del sistema en el reconocimiento de las distintas acciones.

En la figura 1.3 se muestra con mayor detalle los componentes que forman el sistema de reconocimiento de acciones implementado.

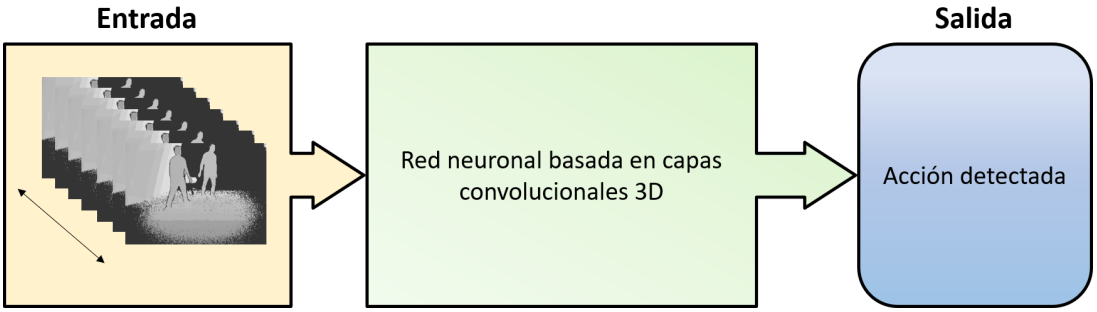



Figura 1.3: Detalle del sistema implementado.

El elemento fundamental del sistema es una red neuronal formada por capas convolucionales 3D, es decir, realizan la operación de convolución sobre secuencias de video, las cuales son la entrada de la arquitectura propuesta. La red proporciona como salida un vector de dimensión el número de acciones consideradas, indicando la probabilidad de que la acción realizada en el video de entrada corresponda a cada una de las clases previamente definidas.

## 1.3 Organización de la memoria

Esta memoria se organiza en cinco grandes capítulos. En el capítulo 2 se realiza una revisión del estado del arte y de las principales aplicaciones del reconocimiento de acciones. A continuación, en el capítulo 3, se exponen los conocimientos necesarios para la comprensión del desarrollo del sistema. Posteriormente, en el capítulo 4 se explica la arquitectura de la red neuronal implementada, el método de entrenamiento utilizado y la preparación de las bases de datos. Después, el capítulo 5 recoge los principales resultados experimentales obtenidos y finalmente, en el capítulo 6 se incluyen las principales conclusiones del trabajo, así como los posibles líneas de trabajo futuro.

Finalmente, se incluyen los capítulos 7 y 8 que muestran los requerimientos *hardware* y *software* necesarios, así como el presupuesto total empleado para el desarrollo e implementación de este TFG.

# Capítulo 2

# Estado del Arte

*Una máquina puede hacer el trabajo de cincuenta hombres ordinarios. Ninguna máquina puede hacer el trabajo de un hombre extraordinario.*

Elbert Hubbard

## 2.1 Introducción

En este capítulo se exponen los trabajos más relevantes de otros grupos de investigación que permiten el reconocimiento de acciones aplicado a diferentes disciplinas. El concepto de reconocimiento de acciones humanas en el área de la visión artificial ha permitido mejorar las técnicas empleadas en sistemas de vídeo-vigilancia, de conocimiento de la situación, seguridad y hogares inteligentes o *smart homes*. Sin embargo, la detección de acciones es una tarea complicada debido a factores como la propia definición de acción en un determinado contexto y según su aplicación específica, la representación de sus características fundamentales o la base de datos utilizada.

## 2.2 Trabajos previos

En este apartado se presentan diferentes trabajos realizados por la comunidad científica en el área del reconocimiento de acciones, organizados en función de sus diferentes aplicaciones, ya introducidas en el capítulo 1.

1. **Aplicaciones a la vida activa y asistida en hogares inteligentes:** los avances en las tecnologías modernas han proporcionado la oportunidad de mejorar la calidad de vida de personas mayores o diversos funcionales en un entorno interior inteligente. Las técnicas de reconocimiento de acciones empleadas permiten la monitorización y asistencia de los residentes para asegurar su seguridad y bienestar. El empleo de sensores que proporcionan información visual, ha permitido lograr tareas tales como el reconocimiento de acciones aplicadas a tareas físicas realizadas por pacientes con demencia [33] o el análisis de acciones cotidianas en un entorno interior [34]. En la figura 2.1 se muestra un ejemplo de base de datos utilizada en un entorno interior con el objetivo de detectar las acciones cotidianas que se realizan en el hogar.


Figura 2.1: Ejemplo de imágenes pertenecientes a una base de datos que permite el reconocimiento de acciones humanas cotidianas en un entorno interior [1].

2. **Aplicaciones de monitorización sanitaria:** el objetivo que persiguen los investigadores en este contexto es intentar mejorar la monitorización sanitaria existente, que permita manejar situaciones médicas urgentes y acortar los tiempos de estancia en hospitales.

   A pesar de que las propuestas basadas en sensores visuales no son muy populares en el entorno de la monitorización médica, han sido ampliamente empleadas en implementaciones de sistemas de detección de caídas, especialmente en pacientes que sufren enfermedades degenerativas tales como Alzheimer o demencia [35]. En la figura 2.2 se muestra un ejemplo de base de datos utilizada en un entorno hospitalario con el objetivo de detectar los movimientos de un paciente.

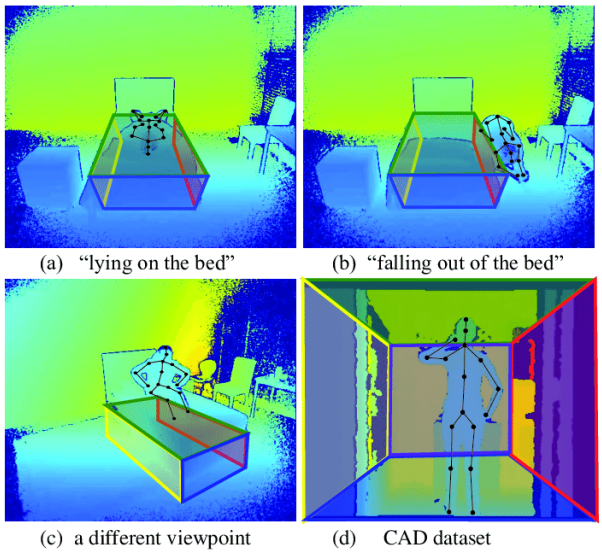
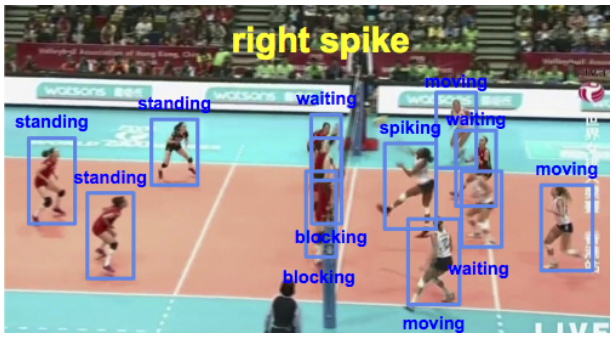


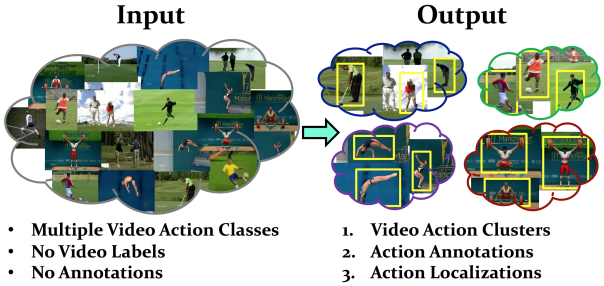

Figura 2.2: Ejemplo de imágenes pertenecientes a una base de datos para el reconocimiento de acciones en un entorno hospitalario [1].

3. **Deportes y acciones al aire libre:** la visión artificial también se emplea para reconocer actividades deportivas que permitan mejorar el rendimiento de los jugadores en los diferentes deportes de interés. Algunas propuestas se basan únicamente en la detección de acciones centradas en un deporte específico [2] y otras en la clasificación entre distintas disciplinas [3]. En la figura 2.3 se muestran ejemplos de reconocimiento de acciones en los casos anteriormente comentados.

4. **Aplicaciones de seguridad y video-vigilancia:** tradicionalmente la video-vigilancia ha sido realizada por un operador, observando continuamente el comportamiento de las personas capturadas




(a) Ejemplo de clasificación de acciones en el ámbito de un mismo deporte, en concreto el volleyball.




(b) Ejemplo de clasificación de deportes de distintas disciplinas.

Figura 2.3: Ejemplo de reconocimiento de acciones en un deporte específico (volley) 2.3a [2] y clasificación entre distintas disciplinas 2.3b [3].

por las cámaras. Un número elevado de cámaras y, por tanto, un incremento de los vídeos a observar simultáneamente, provoca la necesidad de una mayor concentración por parte de los operarios al realizar su desempeño, incrementando su estrés y resultando en una disminución de sus niveles de productividad. La seguridad ha encontrado en la visión artificial una herramienta muy útil para la detección de anomalías en el comportamiento humano en entornos interiores y exteriores, tal y como se observa en [4, 36]. En la figura 2.4 se muestra la detección de acciones aplicada a las personas en el metro.

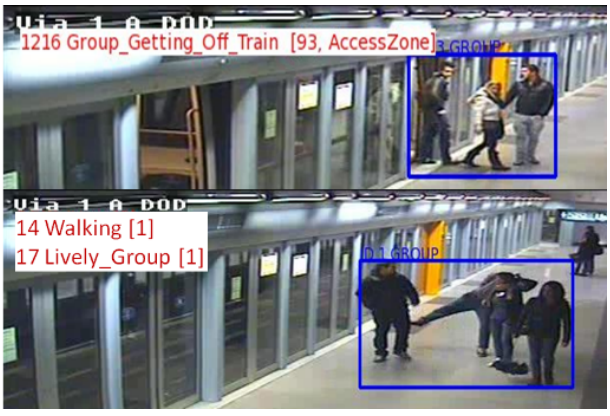



Figura 2.4: Ejemplo de aplicación del reconocimiento de acciones a la seguridad en un lugar público [4].

El sistema de detección de acciones desarrollado en este TFG se basa en el empleo de *Deep Learning*, por lo que se debe realizar una breve revisión de las técnicas empleadas en este campo. Tal y como se expuso en el capítulo 1, una gran parte de los trabajos, se basan en el empleo de secuencias de

imágenes RGB o RGB-D, realizando la detección de acciones mediante arquitecturas basadas en capas convolucionales 3D [37, 38] o LSTM [39, 40].

En el caso de emplear únicamente información de profundidad, existen muy pocos artículos publicados que empleen redes neuronales como método de clasificación. Entre ellos, destacan los trabajos [41] y [42].

Para un mayor detalle del estado del arte en el reconocimiento de acciones basado en *Deep Learning* se puede consultar [43].

# Capítulo 3

# Estudio teórico

*Y así, del mucho leer y del poco dormir, se le secó el cerebro de manera que vino a perder el juicio.*

Miguel de Cervantes Saavedra

## 3.1 Introducción

En este capítulo se aborda el estudio de los conceptos teóricos necesarios para la realización de este trabajo. En concreto se analizan la adquisición y procesado de la información de profundidad, y se realiza una revisión de las técnicas utilizadas en la implementación de un sistema basado en Redes Neuronales (NNs), incluyendo tanto las capas de las que está formado, como las operaciones necesarias para su correcto funcionamiento.

## 3.2 Imágenes de profundidad

En la actualidad, las cámaras más conocidas son las de color o *RGB*. Estas cámaras permiten la obtención de imágenes de color que caracterizan los objetos capturados en la escena. Sin embargo, presentan una serie de inconvenientes a tener en cuenta:

- **Sensibles a la iluminación.** Las cámaras de color no presentan invarianza ante la iluminación, es decir, las imágenes proporcionadas son sumamente dependientes del nivel de luz. Por ello, requieren un mínimo en la iluminación de la escena para ofrecer buenos resultados limitando así su empleo a entornos bien iluminados.
- **Sin información espacial del entorno.** Una única cámara RGB no permite obtener información espacial del entorno. Con el empleo conjunto de dos cámaras es posible conseguir dicha información, pero a expensas del aumento de complejidad del sistema de visión.
- **Problemas con la privacidad.** Este tipo de cámaras obtienen una información que permite el reconocimiento de las personas en la escena, pudiendo presentar problemas en aplicaciones con restricciones de privacidad marcadas por la Ley Orgánica de Protección de Datos de Carácter Personal (LOPD) vigente.

En los últimos años, han aparecido las cámaras RGB-D [20] que, además de la información de color, proporcionan un canal con información de profundidad (distancia de cada punto de la escena a la cámara). Estas cámaras, permiten obtener información espacial del entorno, pero siguen teniendo problemas relacionados con la privacidad de las personas si se utiliza la información de color. Por ello, el sistema desarrollado para la detección de acciones en este TFG se basa únicamente en imágenes de profundidad que permiten subsanar dichos inconvenientes.

Las imágenes de profundidad (*Depth images*) proporcionan información de la distancia de los objetos de la escena con respecto al sensor, ya que el valor de cada píxel representa la distancia de cada punto de la escena al propio sensor. Para la adquisición de estas imágenes pueden emplearse cámaras propiamente de profundidad (2.5D), entre las que se incluyen las cámaras RGB-D.

En la figura 3.1 se muestra un ejemplo de imagen de profundidad en la cual se observa a través de una escala de colores, la distancia de cada punto de la escena con respecto al sensor de la cámara. Los valores representados en dicha escala se encuentran en milímetros y se debe tener en consideración que la imagen ha sufrido un procesado para eliminar los valores de píxel superiores a los 4 metros, sustituyendo su valor por un 0. Esto se ha hecho debido a que las medidas obtenidas a distancias superiores a 4 metros, presentan una gran cantidad de ruido.

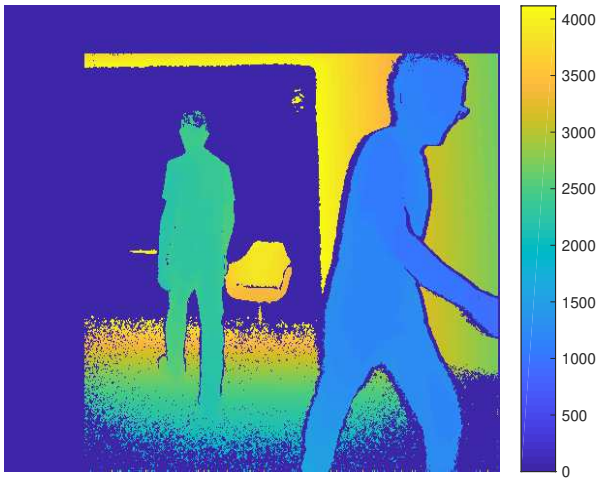



Figura 3.1: Ejemplo de imagen de profundidad.

Existen múltiples técnicas para la obtención de información de profundidad, entre las que destacan los sensores de profundidad basados en visión estéreo, basados en luz estructurada y los basados en tiempo de vuelo (*ToF*). A continuación se realiza una descripción de los fundamentos de medida de las alternativas mencionadas.

### 3.2.1 Sensores de profundidad basados en visión estereoscópica

Los sensores de profundidad basados en visión estereoscópica son ampliamente utilizados en el área de la visión artificial. La obtención de información de profundidad por parte de las cámaras estéreo se basa en el empleo de algoritmos de visión artificial inspirados en la visión binocular humana. Un ejemplo de una posible configuración de cámaras estéreo se muestra en la figura 3.2. Para un mayor detalle se puede consultar el trabajo [44].

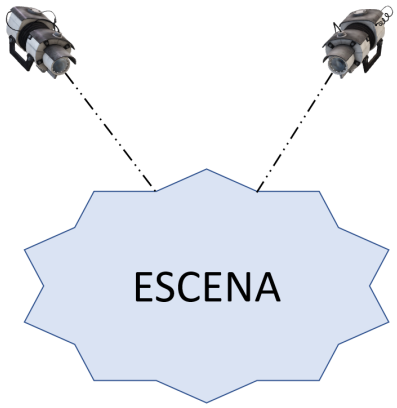



Figura 3.2: Ejemplo de configuración de un sistema de cámaras estéreo.

Una vez capturadas las dos imágenes en paralelo desde dos puntos de vista diferentes, se calcula la información de profundidad estimando las disparidades en los puntos más significativos de ambas imágenes tal y como se muestra en la figura 3.3.

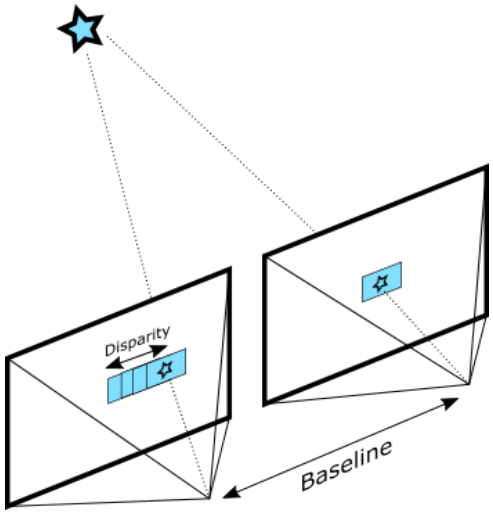
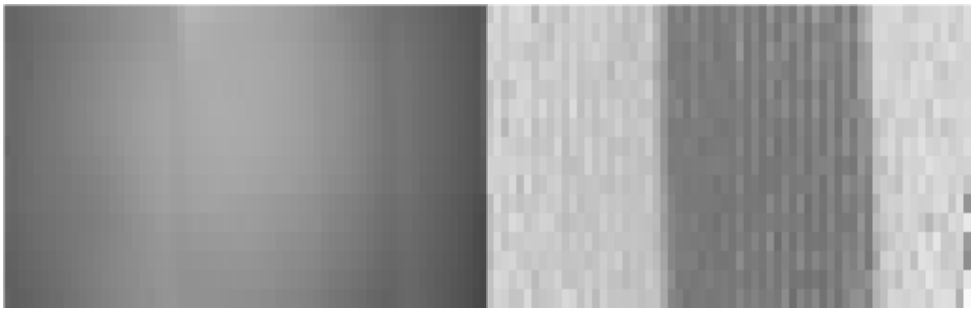



Figura 3.3: Ejemplo de obtención de información de distancia a partir de las disparidades encontradas entre imágenes capturadas con una cámara estéreo [5].

La mayor problemática de esta técnica está asociada a la correspondencia de puntos entre el par de imágenes presentando las siguientes desventajas:

- **Coste computacional elevado.** El cálculo de las disparidades requiere un algoritmo que supone un coste elevado con respecto a las otras técnicas que se comentan en apartados posteriores ( 3.2.2 y 3.2.3).
- **Necesidad de sincronización.** Las cámaras estéreo requieren una sincronización para capturar la escena al mismo tiempo en ambos extremos para evitar posibles inconsistencias en la correspondencia de puntos en situaciones de movimiento.
- **Problemas de textura.** Este tipo de cámaras pueden sufrir problemas en la obtención de la información de distancia de objetos cuyos valores de nivel de gris son iguales con respecto a otros

elementos de la escena, siendo muy difícil distinguir si se encuentran a diferente distancia de la cámara. Este suceso se observa en la figura 3.4 donde se compara una imagen capturada con una cámara estéreo y una cámara ToF de un objeto situado delante de una pared con un nivel de gris similar. Este problema tiene como solución la implementación de un sistema estéreo que contenga iluminación externa, empleando diferentes patrones de luz, para forzar la creación de texturas artificiales y generación de sombras que permitan una correcta correspondencia de puntos y evitar el problema comentado.



(a) Imagen tomada con cámara estéreo. (b) Imagen tomada con cámara ToF.

Figura 3.4: Comparación del problema de textura entre una cámara estéreo y una cámara ToF [6]. En ambas figuras se muestra la situación de un objeto situado delante de una pared, cuyos niveles de gris son similares al fondo.

En la figura 3.4a se observa como la cámara estéreo, sin iluminación externa, es incapaz de diferenciar que el objeto se encuentra a diferente distancia que la pared. Sin embargo, en la figura 3.4b, la cámara ToF emplea otra técnica para la obtención de la información de profundidad que permite una correcta diferenciación entre la distancia al objeto y al fondo. Esta técnica se explica en detalle en el apartado 3.2.3.

### 3.2.2 Sensores de profundidad basados en luz estructurada

Los sensores de profundidad de luz estructurada se basan en el estudio de la deformación que sufre un patrón de luz conocido proyectado sobre la escena. Esto supone una mejora considerable con respecto a la cámara estéreo clásica, ya que permite evitar el problema de textura comentado en la sección 3.2.1. La información de distancia se obtiene a partir de dicha deformación provocada por los cuerpos y objetos de la imagen.

Normalmente el patrón empleado es aleatorio y su objetivo es aumentar la textura de la escena de cara a obtener correctamente las correspondencias en el par de cámaras estéreo. La luz estructurada puede ser de dos tipos, visible o invisible al ojo humano, siendo la primera de estas de gran aplicación en el escaneo tridimensional, la virtualización de espacios, etc. El segundo tipo de luz estructurada se emplea mediante patrones de luz infrarroja o de luz a alta frecuencia. Este es el caso de la *Kinect v1* de *Microsoft* [45], la *Xtion-Pro* de *Asus* [46] o la *RealSense* [47] de *Intel*. En este caso la cámara empleada posee tanto un proyector del patrón como un sensor infrarrojo tal y como se observa en la figura 3.5.

Generalmente las cámaras basadas en luz estructurada proporcionan junto con las imágenes de profundidad, imágenes de color o RGB, es decir, son del tipo RGB-D. Sin embargo, se deben mencionar ciertas limitaciones, tales como la dependencia a la iluminación o el rango de distancias a medir, debido principalmente, a la necesidad de ver el patrón proyectado por parte del sensor infrarrojo. Si lo anterior no ocurre, la triangulación no se realiza correctamente y por tanto, la cámara ofrece una imagen con medidas con una precisión menor, funcionando como una cámara estéreo normal. Por último, es destacable


Figura 3.5: Cámara Xtion Pro de Asus. [7]

el ruido que introducen este tipo de cámaras debido a las oclusiones, dado que el receptor infrarrojo tiene un campo de visión diferente al emisor del patrón al situarse a cierta distancia.

### 3.2.3 Sensores de profundidad basados en tiempo de vuelo

Los sensores de profundidad basados en tiempo de vuelo o ToF son ampliamente conocidos en la actualidad, debido principalmente a su gran versatilidad, velocidad y fiabilidad en la obtención de información de profundidad. El ejemplo más conocido y empleado de cámara ToF es la segunda versión de Kinect (*Kinect v2*) de *Microsoft* [21].

En la figura 3.6 se muestra la cámara Kinect v2, en la que se aprecian sus partes más representativas. Esta incorpora una cámara de color de alta resolución, junto con un sensor de profundidad ToF y un array de emisores infrarrojos para el mismo.

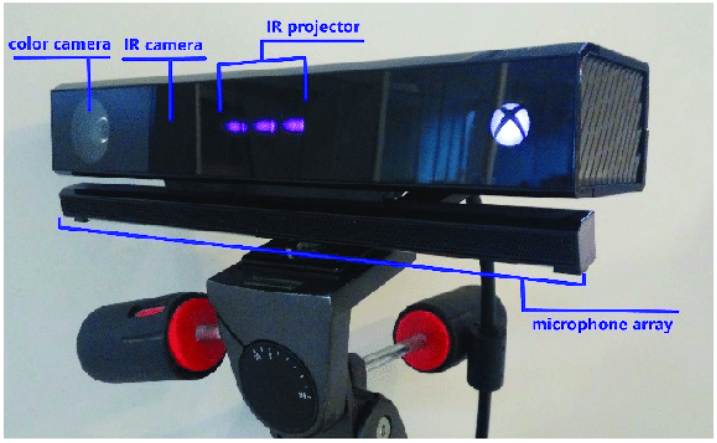



Figura 3.6: Cámara Kinect v2 de Microsoft. [8]

La obtención de la información de distancia de cada punto de la escena al sensor se resuelve mediante la medida del tiempo de vuelo de una señal emitida previamente. Debido a que habitualmente el tiempo de vuelo es muy pequeño, su medida se realiza de forma indirecta. Una de las tecnologías más importantes que se ha desarrollado basándose en el principio ToF son las fuentes de luz moduladas con detectores de fase, consistentes en la emisión de una portadora con una frecuencia de modulación conocida y posteriormente,

la medida del desplazamiento en fase entre la señal emitida y la recibida. En la figura 3.7 se muestra el principio de funcionamiento de una cámara ToF empleando modulación continua basada en la detección de fase de una portadora sinusoidal.

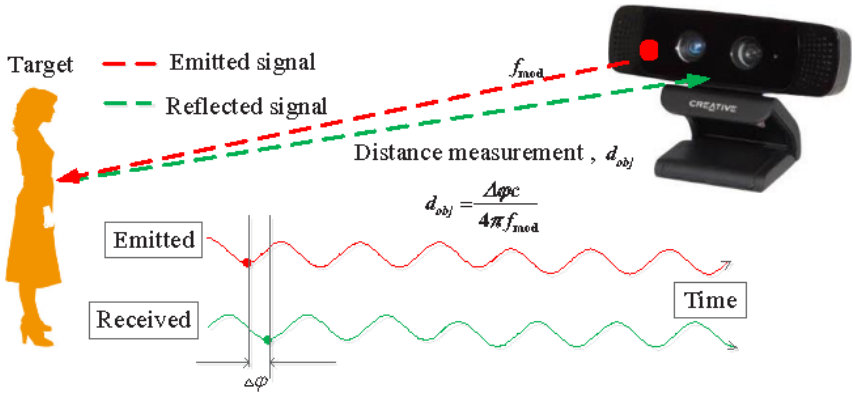



Figura 3.7: Ejemplo de obtención de información de distancia empleando modulación continua y detectores de fase [9].

Seguidamente se expone una explicación más detallada de la técnica ToF apoyándose en la figura 3.8, que ejemplifica una medida de distancia basada en la emisión de una señal pulsada. En la misma se observa la señal infrarroja radiada por el emisor y la recibida captada por el sensor de profundidad fruto de la reflexión de la primera.

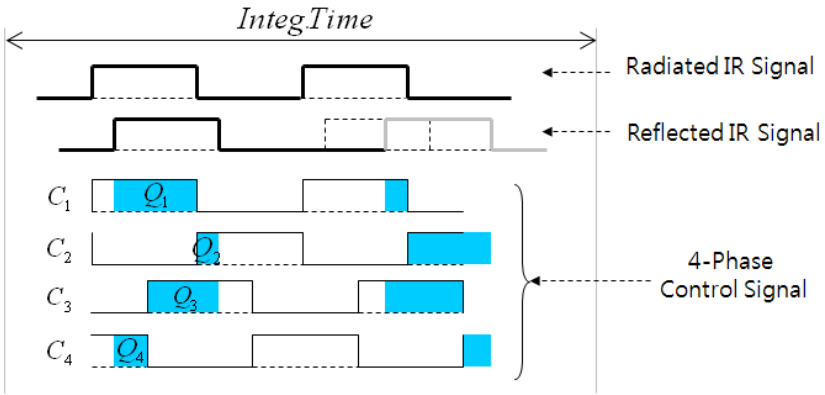



Figura 3.8: Representación de la señal emitida, recibida y las señales de control (C1-C4) con sus respectivos cuantos de carga(Q1-Q4) [10].

El análisis del desplazamiento entre ambas durante un periodo T, denominado periodo de medida, permite obtener la información de distancia. Esto se consigue dividiendo el periodo T en cuatro desplazamientos de 90º dando lugar a las señales de control (C1-C4) mostradas en la figura 3.8. La diferencia de fase entre el haz emitido y el rebotado se obtiene a partir de la relación establecida por la carga acumulada en los transistores MOSFET que forman el sensor (Q1-Q4) tal y como se define en la ecuación 3.1 siendo $\alpha$ la diferencia de fase. Estas medidas se repiten un número elevado de veces y se acumulan en el denominado tiempo de integración. La finalidad de realizar múltiples medidas es el aumento en la precisión de las mismas .

$$\alpha = arctan(\frac{Q_3 - Q_4}{Q_1 - Q_2}) \tag{3.1}$$

Una vez obtenida la diferencia de fase entre el haz emitido y recibido se puede estimar la distancia a través de la velocidad de la luz $c$, la frecuencia de modulación de la onda emitida $f_{mod}$ y el desfase calculado $\alpha$:

$$d = \frac{c}{2f_{mod}} \frac{\alpha}{2\pi} \tag{3.2}$$

De la ecuación anterior se deduce la distancia máxima ($d_{MAX}$) que puede medir una cámara ToF sin errores. Esta distancia se corresponde con una diferencia de fase entre señales de $2\pi$ ya que superando ese valor, es imposible determinar si el desfase entre los haces de luz es de $\alpha$ o $\alpha + n2\pi$.

$$d_{MAX} = \frac{c}{2f_{mod}} \tag{3.3}$$

Finalmente, cabe destacar que las cámaras ToF presentan diferentes fuentes de errores de medida de profundidad, que pueden afectar a la precisión de los datos proporcionados. Las más importantes se listan a continuación:

- **Interferencia por multicamino.** Este tipo de interferencias son provocadas por las múltiples reflexiones que sufren las ondas infrarrojas en situaciones como bordes o esquinas. Lo anterior tiene como consecuencia la llegada al sensor de varias ondas reflejadas, la directa (una única reflexión) junto con las que han sufrido dos o más reflexiones. Esto induce a medidas erróneas que conducen a la introducción de ruido y a una posible sobre-estimación de la distancia. Un ejemplo del efecto de la interferencia por multicamino se refleja en la figura 3.9. Cabe destacar, que los errores por interferencia multicamino pueden modelarse y corregirse [48], disminuyendo su efecto sobre las medidas de distancia.

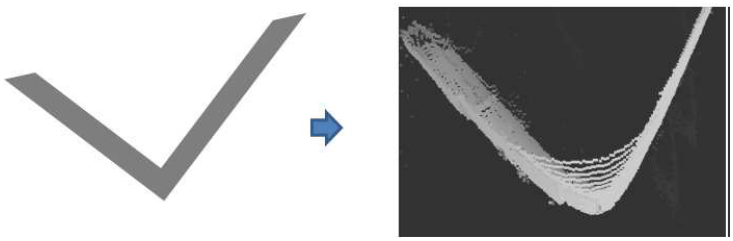


Figura 3.9: Ejemplo del problema del multicamino en una esquina [10].

- **Artefactos de movimiento.** Este tipo de cámaras requieren un determinado tiempo de integración que permita la obtención de la diferencia de fase entre la señal emitida y recibida. Si en dicho intervalo de tiempo, se producen grandes cambios en la escena, se registran medidas ruidosas, es decir, erróneas. En la figura 3.10 se muestra un ejemplo del problema expuesto, donde se observa como el punto rojo se ha desplazado en el intervalo de integración. Esto es debido a que se toman medidas que corresponden al objeto, y otras que son parte del fondo, por lo que al realizar la integración, se obtiene una imagen de profundidad con valores intermedios entre ambas, manifestándose como un suavizado alrededor del objeto. Al igual que en el caso anterior, existen técnicas que permiten reducir el efecto de los artefactos de movimiento sobre las medidas de de profundidad [49].

- **Errores por iluminación no uniforme.** Las cámaras ToF utilizan potentes módulos de iluminación infrarroja cuyo mayor inconveniente es proporcionar una iluminación muy centrada. Esto provoca una falta de iluminación en las esquinas de la imagen y como consecuencia errores en la medida de distancia tal y como se observa en la figura 3.11.

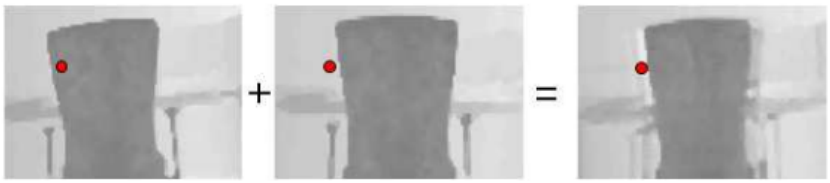


Figura 3.10: Ejemplo del efecto de los artefactos de movimiento en la medida de distancia [10].

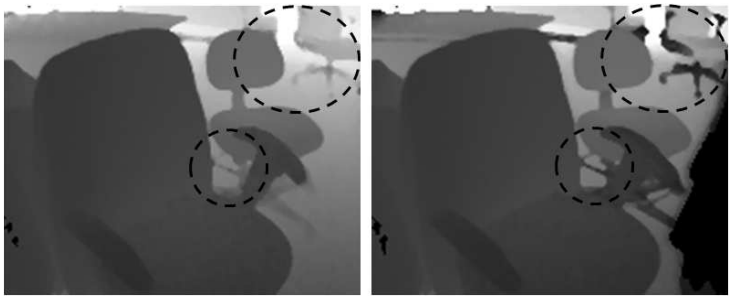


Figura 3.11: Ejemplo del problema de distorsión provocado por la iluminación no uniforme [10].

## 3.3 Redes Neuronales Artificiales

Las NNs se tratan de un modelo de procesamiento de información inspirado en la forma biológica de trabajar del cerebro humano como unidad de procesamiento. El elemento fundamental de estos modelos es su estructura compuesta por un gran número de elementos de procesamiento denominados *neuronas* que trabajan conjuntamente para lograr un objetivo común. Son configuradas para desempeñar todo tipo de aplicaciones específicas a través de una etapa de entrenamiento. El modelo de una neurona artificial clásica consta de los siguientes elementos:

- Un conjunto de **entradas** $x_1$, $x_2$, $x_n$ que portan la información del exterior.
- Un conjunto de **pesos** $w_1$, $w_2$, $w_n$ correspondientes a cada entrada. Estos pesos son coeficientes que definen la importancia de cada neurona dentro de la función de agregación. La actualización de los pesos se realiza en la etapa de entrenamiento para conseguir el propósito marcado.
- Una **función de agregación** que a partir de las entradas con sus correspondientes pesos obtiene un valor de procesamiento. La operación más común es la suma ponderada de las entradas con sus pesos.
- Una **función de activación** encargada de transformar el valor de procesamiento obtenido por la función de agregación, en un valor de activación, normalmente acotado.
- Una **salida** que proporciona el resultado obtenido en la función de activación.

En la figura 3.12 se muestra un esquema gráfico de los elementos de una neurona artificial comparándolos con los de una neurona biológica.

El proceso de aprendizaje de las NNs consiste en la actualización de los pesos correspondientes a cada neurona. Existen tres tipos de aprendizaje que se exponen a continuación:

- **Aprendizaje supervisado.** En este tipo de aprendizaje se realiza el entrenamiento de las NNs a partir de datos etiquetados, intentando lograr, dados unos datos de entrada, los correspondientes datos de salida etiquetados. El entrenamiento supervisado es útil y ampliamente utilizado en tareas

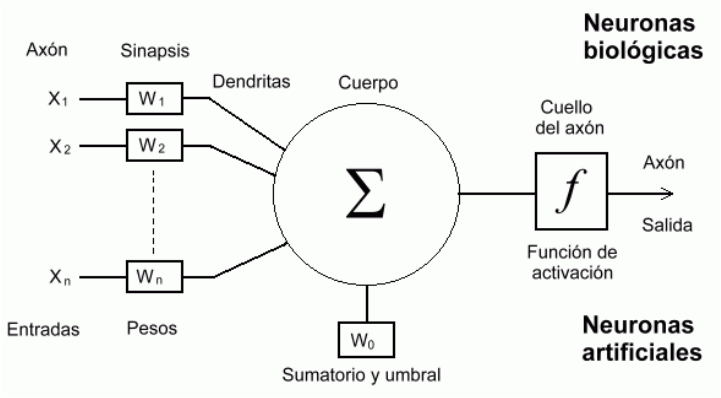



Figura 3.12: Comparación de los elementos de una neurona artificial y biológica [11].

de clasificación y regresión, sin embargo, uno de sus principales inconvenientes es la necesidad de grandes bases de datos que permitan obtener buenos resultados.

- **Aprendizaje no supervisado.** El entrenamiento se lleva a cabo con un conjunto de datos no etiquetados. En contraposición al aprendizaje supervisado, los algoritmos no conocen la relación de los datos de salida con los de entrada, por lo que trata de clasificarlos por sí mismos mediante restricciones impuestas a través de la función de pérdidas y métricas indirectas.
- **Aprendizaje semi-supervisado.** Se basa en la creación de un modelo a partir de la exploración de datos, desarrollando múltiples acciones de prueba y error para lograr un aprendizaje por sí mismo apoyado en el empleo de recompensas y penalizaciones.

Las neuronas pertenecientes a las NNs se organizan en capas o *layers*. Cada una de ellas están formadas por un conjunto de neuronas cuyas entradas proceden de la misma capa o de una diferente y proporcionan unas salidas que a su vez son las entradas de las siguientes capas. De forma general se consideran tres grandes tipos de capas:

- **Capa de entrada:** recibe los datos de entrada.
- **Capas ocultas:** son aquellas encargadas de las operaciones necesarias para lograr el ajuste de parámetros que permita conseguir el objetivo marcado.
- **Capa de salida:** se encarga de proporcionar la salida de la red en función de los datos de entrada introducidos.

En función del número de capas de las NNs se habla de red monocapa o también denominada *perceptrón* si únicamente contiene la capa de entrada y la de salida, siendo esta última la encargada en la tarea del procesado de los datos. Por otro lado se denomina red multicapa o *perceptrón multicapa* cuando, además de contener una capa de entrada y de salida, se añaden capas ocultas tal y como se muestra en la figura 3.13.

Para un mayor detalle con respecto a las NNs clásicas se recomienda consultar [50].

### 3.3.1 Redes neuronales convolucionales

Las Redes Neuronales Convolucionales (CNNs) son un tipo de redes neuronales basadas en el empleo de operaciones de convolución sobre los píxeles de una imagen con el objetivo fundamental de funcionar

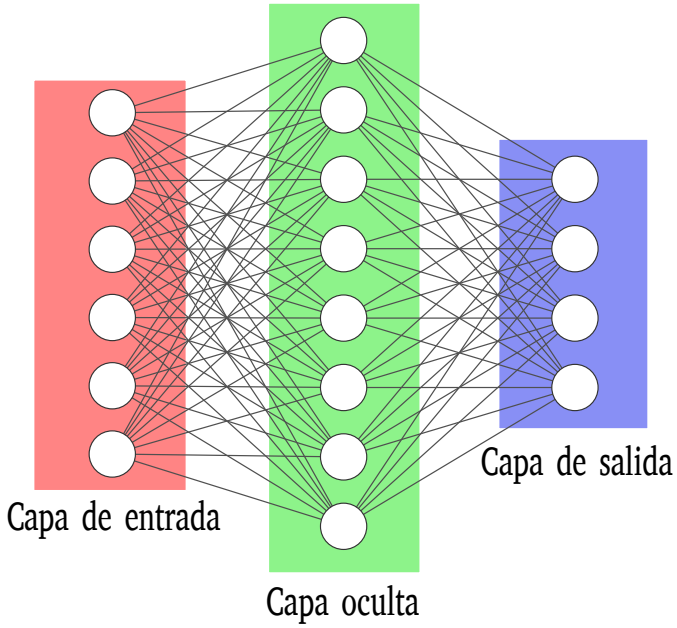



Figura 3.13: Ejemplo de perceptrón multicapa.

de forma similar a las neuronas que actúan sobre la corteza visual primaria del cerebro humano. La evolución y el aumento en la velocidad en el cálculo y procesamiento de datos gracias a la aparición de las Graphics Processing Unit (GPU) ha permitido el desarrollo de NNs más complejas que permiten trabajar con imágenes e incluso vídeos en tiempo real.

Las CNNs están formadas por capas de filtros convolucionales de varias dimensiones (1, 2 o 3 dimensiones) que realizan las operaciones sobre los píxeles de las imágenes de entrada con un tamaño concreto. Cada capa anterior, viene complementada de una función no lineal de activación de salida que proporciona una matriz de salida a la siguiente capa y otorga no linealidades al modelo. En general, las CNNs constan de 2 etapas bien diferenciadas:

1. **Extracción de características:** se trata de la fase inicial de una arquitectura de red neuronal cuya tarea es la extracción de características a través de las capas convolucionales. Según se avanza en el número de capa de la red, se decrementa la reacción de las neuronas ante variaciones de las imágenes de entrada y mayor es la abstracción de estas, reconociendo cada vez información más compleja tal y como se muestra en la figura 3.14.

2. **Clasificación y/o regresión:** Se trata de la fase final de la arquitectura de la red. Se constituye por las denominadas capas de densas o *fully connected* formadas por neuronas convencionales conectadas entre sí que a partir de la información obtenida en la etapa de extracción de características, realizan la clasificación o regresión de los datos en función del objetivo marcado.

A continuación se comentan las operaciones y capas más importantes necesarias para la comprensión del sistema de detección de acciones implementado en este TFG.

#### 3.3.1.1 Capas convolucionales

Las capas convolucionales realizan operaciones sobre los píxeles de las imágenes de entrada aplicando filtros discretos o *kernels*. Un *kernel* se define como un parche que delimita el área de la imagen sobre la que se realizarán determinadas operaciones. La aplicación de la operación de convolución consiste en el

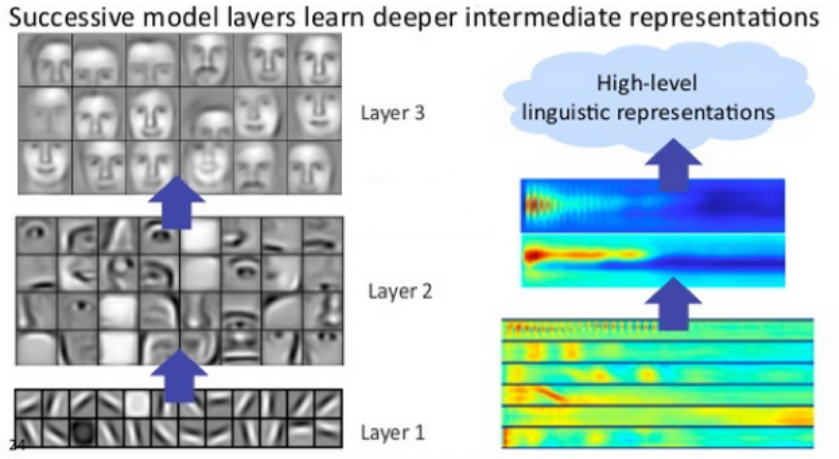



Figura 3.14: Ejemplo de extracción de características de las distintas capas de una CNN [12].

deslizamiento del *kernel* definido sobre las imágenes de entrada, realizando un conjunto de operaciones, entre las que se incluyen productos y sumas, obteniendo como resultado un mapa de características.

Dichos filtros pueden ser unidimensionales, bidimensionales o tridimensionales, definiendo así una operación de convolución 1D, 2D o 3D respectivamente. En la figura 3.15 se muestra la operación básica de convolución aplicando un filtro bidimensional de 3x3. Se basa en el desplazamiento del kernel sobre la imagen de 8x8, obteniendo el valor del elemento del mapa de características a través del sumatorio del producto de los elementos superpuestos entre la imagen y el filtro. En concreto en este ejemplo, el resultado de realizar la operación anteriormente descrita es un valor de 0.

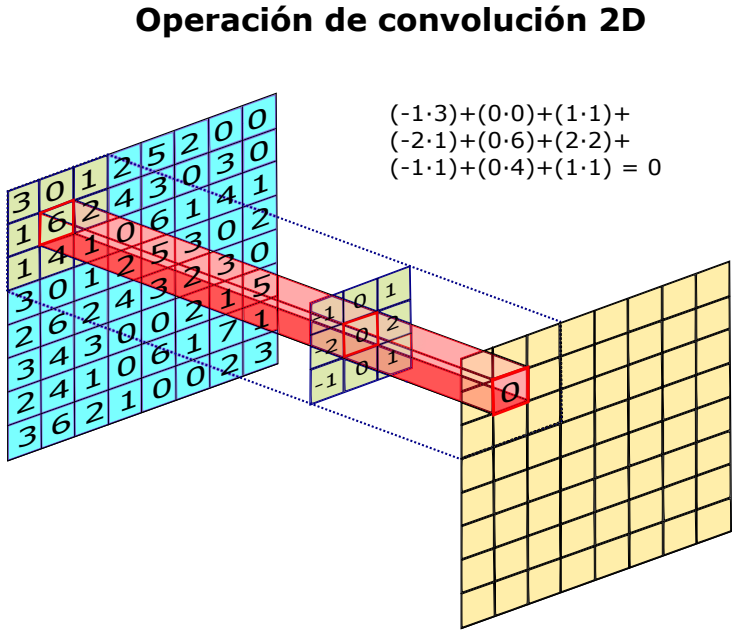



Figura 3.15: Ejemplo de operación de convolución sobre una imagen de entrada aplicando un filtro de tipo Sobel. El filtro de tipo Sobel mostrado se construye a través de la derivada de la gaussiana y es empleado para la obtención de los bordes de una imagen.

Para la consecución del objetivo marcado en el presente TFG es necesaria la obtención de características de secuencias de imágenes de entrada, es decir, además de la información espacial, es imprescindible la información temporal que proporcione una coherencia en la ejecución de las acciones permitiendo su

correcta detección. Es por ello que se requiere la aplicación de filtros de convolución tridimensionales sobre las secuencias de entrada, en las cuales esa tercera dimensión representa el tiempo. La operación de convolución es análoga a la vista en la figura 3.15 pero aplicando la convolución sobre una secuencia de varias imágenes, incluyendo la dimensión temporal, además de las dos dimensiones espaciales. En la figura 3.16 se muestra un ejemplo de la operación de convolución 3D y el aspecto del *kernel* que se aplica.

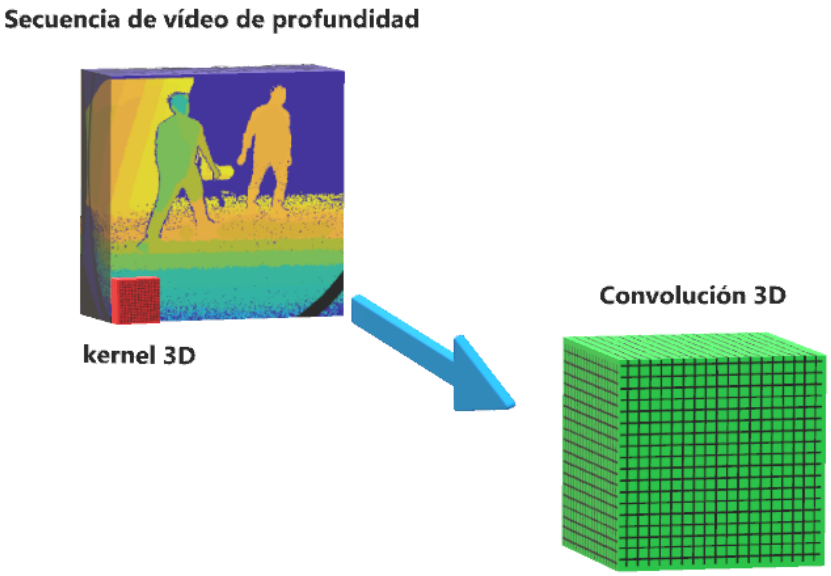



Figura 3.16: Ejemplo de operación de convolución 3D sobre una secuencia de imágenes de entrada de profundidad aplicando un *kernel* tridimensional.

A continuación se explican los principales parámetros necesarios para comprender de una forma exhaustiva las operaciones realizadas por las capas convolucionales de las CNNs:

1. **Tamaño del kernel:** define el tamaño del parche que se aplica sobre la imagen o imágenes de entrada para realizar sobre ellas la operación de convolución. El número de dimensiones de este kernel viene definido por el tipo de convolución a realizar, es decir, para una operación de convolución unidimensional, bidimensional y tridimensional, el kernel está definido por 1, 2 y 3 dimensiones respectivamente.

2. **Stride:** es el desplazamiento, medido en píxeles, que realiza el filtro de convolución en cada deslizamiento. El incremento de este parámetro provoca que el filtro se desplace con un intervalo de píxeles mayor y por tanto, exista menos solapamiento entre las operaciones de convolución.

3. **Padding:** debido a que el tamaño del mapa de características obtenido tras la operación de convolución es siempre menor que la imagen sobre la que se ha aplicado, se debe de incluir una operación adicional para evitar el efecto *shrink* que puede provocar la desaparición del mapa de características tras la aplicación de múltiples operaciones de convolución concatenadas. La solución es el empleo del *padding* que incluye píxeles adicionales, incrementando la dimensión espacial de la entrada, obteniendo tras la convolución un mapa de la misma dimensión que la imagen inicial.

#### 3.3.1.2 Capas de Pooling

Entre las capas convolucionales, es muy común añadir una capa de *pooling*. La función de estas capas es la obtención de las características más importantes, la reducción de dimensionalidad y, por tanto, de

parámetros de la red, así como su coste computacional. La consecuencia de la aplicación de dichas capas es la disminución del tiempo de entrenamiento.

Existen diferentes tipos de capas de *pooling*, entre las que destacan las capas de *Average Pooling*, las cuáles calculan el valor medio entre los valores definidos en función del tamaño del *kernel* y las capas de *Max Pooling*, las cuáles obtienen el valor máximo entre aquellos que se encuentran superpuestos por el *kernel*. En la figura 3.17 se muestra un ejemplo de operación de *pooling* sobre una imagen de 4x4 píxeles aplicando un filtro de 2x2. En concreto, se trata de una operación de *max-pooling* que consiste en obtener el valor máximo de los píxeles encerrados por el filtro de 2x2, el cual recorre toda la imagen.

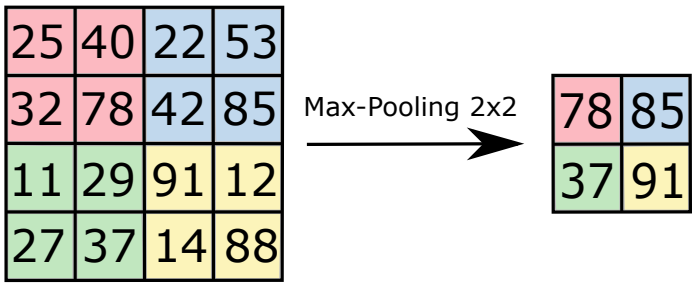



Figura 3.17: Ejemplo max pooling con un filtro de 2x2 píxeles.

#### 3.3.1.3 Capas densas

Este tipo de capas están formadas por las denominadas *neuronas clásicas* utilizadas en los perceptrones multicapa, tal y como se observa en la figura 3.18. El empleo de estas capas suele estar destinado a la etapa de clasificación o regresión de la red neuronal, convirtiendo los mapas de características en valores concretos en función del objetivo final de la red.

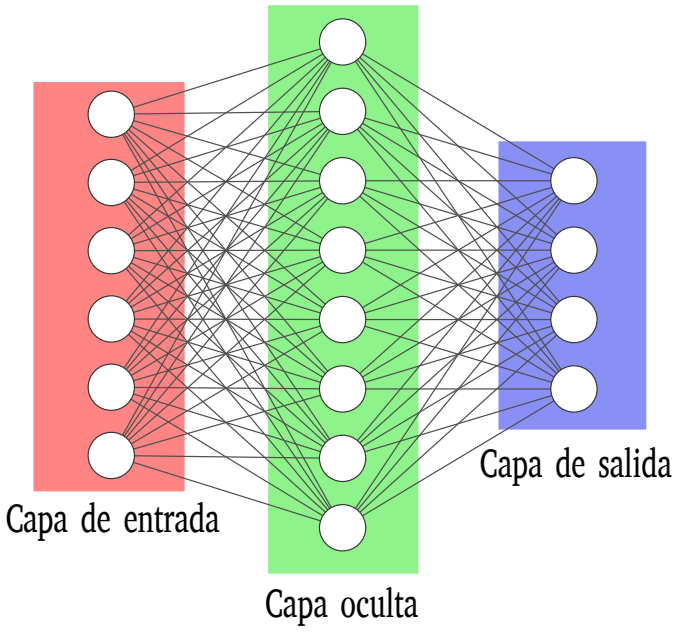



Figura 3.18: Ejemplo de capas de densas.

#### 3.3.1.4 Capas de activación

Son las encargadas de aportar no linealidad a las funciones generadas por las CNNs y de proporcionar una serie de umbrales que sirven para mapear los datos de salida en un espacio de representación distinto al de entrada. La activación puede llevarse a cabo mediante diferentes funciones, como las siguientes:

- **Lineal:** la función lineal representa los datos en un espacio de representación de salida idéntico al de entrada. Es bastante conocida por su empleo en problemas de regresión. En la figura 3.19 se muestra un ejemplo de función de activación lineal.

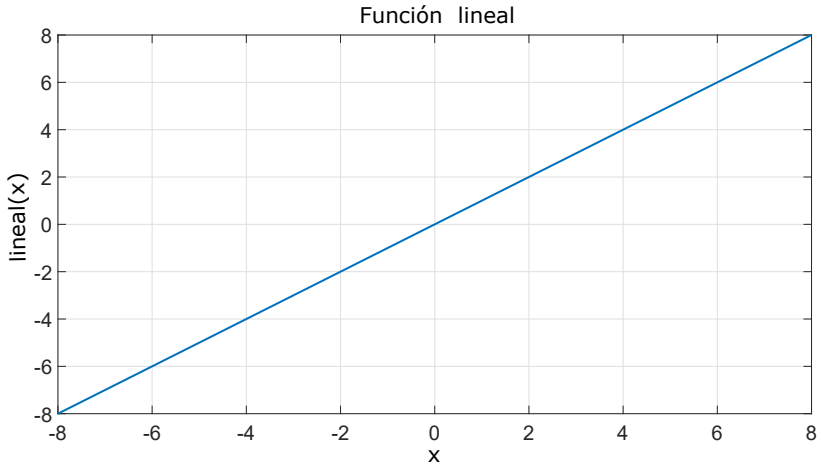



Figura 3.19: Ejemplo de función de activación lineal.

- **Unidad Lineal Rectificada o ReLu:** la función de activación Relu es una de las más empleadas en el diseño de las redes neuronales modernas. Esto es así debido a que facilita el entrenamiento en DNN tal y como se ha demostrado experimentalmente [51]. En la figura 3.20 se muestra un ejemplo de este tipo de funciones de activación. Las principales ventajas que presenta esta función de activación son las siguientes:

  - **Mayor eficiencia computacional:** debido principalmente a su sencillez en el cálculo de la función y sus derivadas asociadas.
  - **Invariante a escala:** el cambio de escala en la entrada se ve reflejado en la parte lineal de la función de activación.
  - **Mejor propagación del gradiente:** se disminuyen los problemas relacionados con el desvanecimiento del gradiente gracias a la ausencia de saturación.

- **Sigmoidal:** la función de activación logística o sigmoidal es una función monótona, diferenciable y dado que ofrece valores entre 0 y 1 se suele emplear para problemas de clasificación. Se puede observar un ejemplo de este tipo de función en la figura 3.21.

- ***SoftMax*:** es una extensión de la función de activación logística empleada para clasificación multiclase. Permite convertir un vector N-dimensional, $S(a)$, de valores reales, en otro vector con las mismas dimensiones, $S$, cuyos valores están normalizados entre 0 y 1. En las expresiones 3.4 y 3.5 se define el procedimiento para la obtención del vector normalizado.

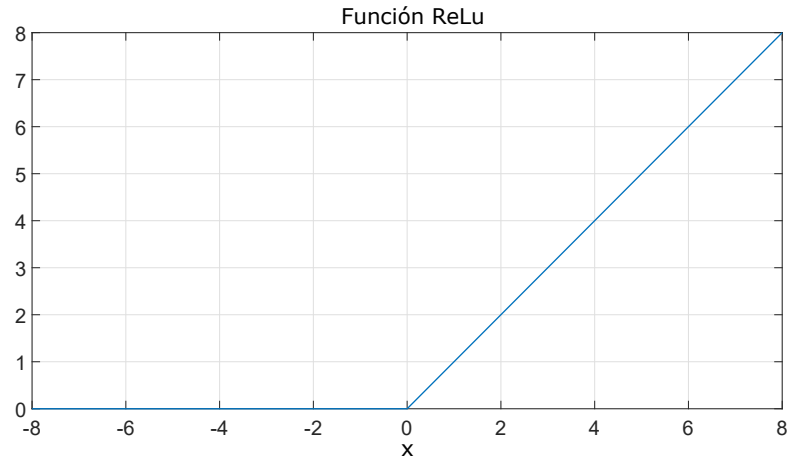



Figura 3.20: Ejemplo de función de activación ReLu.

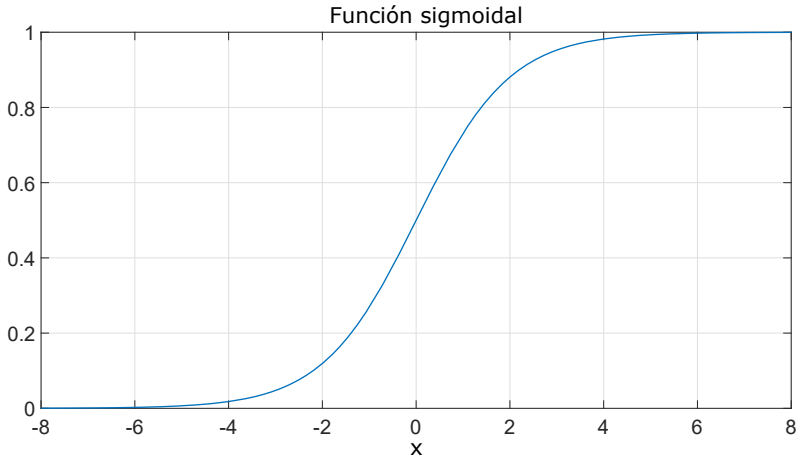



Figura 3.21: Ejemplo de función de activación sigmoidal.

$$S\left(a\right) = \begin{bmatrix} a_1 \\ a_2 \\ ... \\ a_n \end{bmatrix} \rightarrow \begin{bmatrix} S_1 \\ S_2 \\ ... \\ S_n \end{bmatrix} \tag{3.4}$$

$$S_j = \frac{e^{a_j}}{\sum_{k=1}^{n} e^{a_k}} \quad j \in 1..n \tag{3.5}$$

### 3.3.2 Regularización

La regularización consiste en una serie de técnicas empleadas en el contexto del *Machine Learning* con el objetivo de mejorar la generalización en la etapa de entrenamiento de las redes neuronales, es decir, mejorar su comportamiento ante datos que no haya visto antes. A continuación se describen algunas de las técnicas más utilizadas para regularizar:

- **Aumento de datos:** la manera más simple de reducir el sobre-entrenamiento es incrementar el tamaño de los datos para el entrenamiento. Para ello, en el contexto de las CNNs, el aumento de datos consiste en aplicar operaciones tales como desplazamiento, rotaciones, filtrado, escalado...etc. De esta forma se consigue aumentar la generalización del modelo en cuestión. La aplicación de esta técnica no siempre es sencilla, pues se deben tener en cuenta los efectos que puede producir esa modificación de los datos.

- **Dropout:** Es una de las capas de regularización más utilizadas para evitar el sobre-entrenamiento (*overfitting*) cuyo comportamiento y efectos se han estudiado ampliamente [52]. El término *dropout* hace alusión a la eliminación de las contribuciones de ciertas neuronas. La supresión de estas contribuciones se realiza de forma aleatoria en función de una probabilidad. Es importante destacar que las contribuciones únicamente se eliminan durante la etapa de entrenamiento. En la figura 3.22 se muestra un ejemplo de la aplicación de esta técnica.

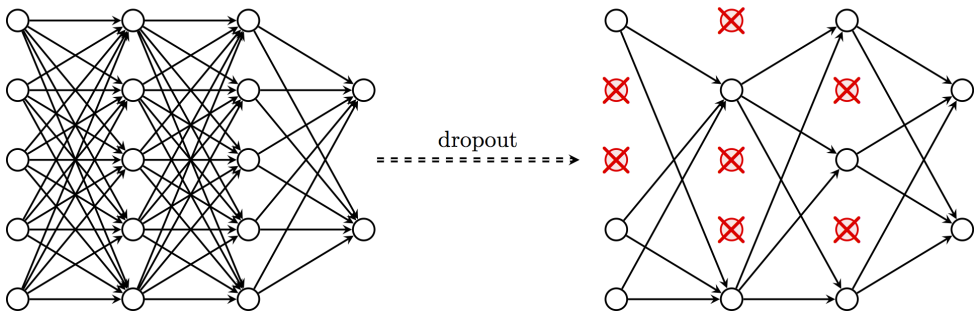



Figura 3.22: Ejemplo de aplicación de técnica de regularización *Dropout*.

### 3.3.3 Función de pérdidas

El empleo de las funciones de pérdidas es una parte fundamental en el contexto de las redes neuronales, pues permite la obtención de métricas para valorar el comportamiento de la red diseñada. Se trata de un valor que advierte sobre el funcionamiento de la red implementada donde la robustez del modelo se incrementa a medida que el valor de función de pérdidas disminuye. A continuación se exponen algunas de las funciones de pérdidas más utilizadas:

- **Error medio cuadrático (MSE):** es ampliamente utilizada en problemas de regresión como la medida de rendimiento. Su definición se muestra en la ecuación 3.6:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(y^{(i)} - y_{pred}^{(i)}\right)^2 \tag{3.6}$$

  donde $y^{(i)} - y_{pred}^{(i)}$ se denomina residuo, y el objetivo de la función de pérdidas MSE es minimizar la suma de los residuos al cuadrado.

- **Error medio absoluto (MAE):** es una métrica que permite determinar el grado de proximidad entre las predicciones y los valores deseados. Su cálculo se muestra en la expresión 3.7:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|y^{(i)} - y_{pred}^{(i)}\right| \tag{3.7}$$

  Tanto MSE como MAE se emplean en modelos predictivos, pero existen diferencias entre ellos que se deben mencionar. MSE tiene buenas propiedades matemáticas para el cálculo del gradiente, por el contrario, MAE tiene problemas de gradiente, asimismo, y con respecto a los casos atípicos o *outliers*, tienen mucha influencia en MSE debido al uso del cuadrado, mientras que MAE presenta una mayor robustez frente a estos.

- **Categorical Cross Entropy (CCE):** la función de pérdidas CCE es comúnmente empleada en problemas de clasificación multi-clase. Mide la divergencia entre dos distribuciones de probabilidad, es decir, si el valor de la métrica es grande, implica que la diferencia entre ambas distribuciones es mayor, y por el contrario, si el valor es próximo a 0, significa que ambas distribuciones son similares entre sí. Esta función de pérdidas viene definida por la ecuación 3.8:

$$CCE = -log\left(\frac{e^{s_p}}{\sum_j^C e^{s_j}}\right) \tag{3.8}$$

  Generalmente, comparada con las funciones de coste cuadráticas, CCE tiene como ventajas una convergencia más rápida y es más común lograr una optimización global.

### 3.3.4 Optimizadores

Los algoritmos de optimización son los encargados de lograr minimizar o maximizar una determinada función marcada como objetivo. En el contexto de las CNNs, se suele referir a minimizar la función de pérdidas escogida para comprobar el funcionamiento del modelo diseñado.

En el ámbito de los optimizadores orientados a las CNNs, se encuentran una gran diversidad de ellos que derivan del clásico descenso por el gradiente estocástico (Stochatic Gradient Descent (SGD)) [53]. En este algoritmo se seleccionan de forma aleatoria pequeños conjuntos de datos de entrenamiento, también denominados *batches* utilizados para la realización de múltiples iteracciones del algoritmo hasta la consecución de la optimización requerida. Su principal parámetro a ajustar es el denominado *learning rate* o ratio de aprendizaje. Se debe seleccionar con sumo cuidado ya que si el *learning rate* es alto, pueden producirse problemas de *overshooting*, es decir, sobreimpulsos en la función de pérdidas durante el entrenamiento. En contraposición, si el *learning rate* es pequeño, el tiempo necesario de entrenamiento es superior pues se realizan muchas iteracciones hasta la convergencia, además, de los problemas asociados a la posible convergencia a un mínimo local.

A partir del SGD, se han desarrollado nuevos algoritmos cuyo *learning rate* es dinámico, es decir, se puede variar durante el entrenamiento. Cabe mencionar como algoritmos adaptativos Adam [54], Nadam [55], Adagrad [56], Rmsprop, Adamax [57]. El presente trabajo se centra en el optimizador Adam por sus características adaptativas y debido a que ha demostrado ser un candidato válido para el entrenamiento de redes neuronales.

El optimizador adaptativo Adam es un algoritmo basado en la optimización de primer orden de la función marcada como objetivo. Se le considera un optimizador adaptativo pues permite la variación de su *learning rate* durante el entrenamiento. El algoritmo requiere 4 parámetros básicos: *lr* o *learning rate*, $\beta_1$ o ratio de caída del primer momento, $\beta_2$ o ratio de caída del segundo momento y $\theta_0$ o vector de parámetros iniciales.

Se debe mencionar que al ser un método adaptativo, el optimizador funciona peor en condiciones de tamaño de *batch* pequeño pues el cálculo de los parámetros necesarios para la obtención del *learning rate* está basado en los propios *batches*. Entre las ventajas de Adam destacan las siguientes:

- Simplicidad en su implementación.
- Computacionalmente eficiente.
- Buen comportamiento ante problemas con gran número de parámetros.
- Mínimos requerimientos de memoria.

A continuación se muestra el pseudocódigo que permite la ejemplificación del optimizador Adam 3.23:

**Algorithm 1.** Adam Optimization Algorithm

**Require:** Objective function $f(\theta)$, initial parameters $\theta_0$, stepsize hyperparameter $\alpha$, exponential decay rates $\beta_1, \beta_2$ for moment estimates, tolerance parameter $\lambda > 0$ for numerical stability, and decision rule for declaring convergence of $\theta_t$ in scheme.

**procedure** ADAM($f, \theta_0$ ; $\alpha$, $\beta_1$, $\beta_2$)
  $m_0$, $v_0$, $t \leftarrow [0, 0, 0]$ # Initialize moment estimates
  # and timestep to zero
  # Begin optimization procedure
  **while** $\theta_t$ has not converged **do**
    $t \leftarrow t + 1$ # Update timestep
    $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ # Compute gradient of objective
    $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ # Update first moment estimate
    $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot (g_t \odot g_t)$ # Update second moment estimate
    $\widehat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ # Create unbiased estimate $\widehat{m}_t$
    $\widehat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ # Create unbiased estimate $\widehat{v}_t$
    $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m}_t / (\sqrt{\widehat{v}_t} + \lambda)$ # Update objective parameters
  **return** $\theta_t$ # Return final parameters

Figura 3.23: Pseudocódigo optimizador adaptativo Adam.

## 3.4 Parámetros de entrenamiento

El entrenamiento es la etapa fundamental en la implementación de un sistema basado en redes neuronales, pues en función de su desempeño, se conseguirán o no los objetivos marcados. Una vez escogidos la función de pérdidas y el optimizador encargado de minimizar su valor, se deben decidir los parámetros fundamentales que definen la etapa de entrenamiento, los cuáles se exponen a continuación:

- **El tamaño de *batch*:** los *batches* se definen como un subconjunto de muestras de entrenamiento cuyo tamaño es sumamente importante para el desarrollo de esta etapa, ya que permite la actualización de los parámetros internos de la red. Una vez que la red ha procesado el *batch* correspondiente, las predicciones obtenidas se comparan con las variables de salida esperadas, calculando el error entre ambas a través de la función de pérdidas elegida para el problema concreto. Seguidamente, el optimizador se encarga de minimizar dicha función. El estudio del comportamiento de la etapa de entrenamiento en función del tamaño de *batch* elegido, se realiza de forma empírica y depende fuertemente del objetivo marcado para la red. Si el tamaño de *batch* es demasiado elevado, puede producirse una baja generalización. Por otro lado, con un tamaño de *batch* demasiado pequeño se puede producir sobre entrenamiento sobre esas muestras, pudiendo entrar en un régimen oscilatorio. Por otra parte, con el empleo de un tamaño de *batch* medio-bajo se ha observado una convergencia más rápida a buenas soluciones [58] ya que, de forma intuitiva, el modelo empieza a entrenar antes de haber visto la totalidad de los datos, permitiendo una mayor generalización. Sin embargo, no se garantiza converger en un mínimo óptimo de la función de pérdidas.

- **El número de épocas:** el concepto de época se denomina al momento en el que todo el set de entrenamiento ha sido pasado a través de la red, es decir, cada muestra de datos ha tenido al menos la oportunidad de modificar los parámetros internos del modelo. En general, el número de épocas suele ser elevado con el objetivo de minimizar lo máximo posible la función de pérdidas calculada y, de esta forma, conseguir la mayor precisión posible en el objetivo marcado. Lo anterior está ligado al punto de divergencia entre el aprendizaje del set de entrenamiento y de validación, siendo el punto en el que ambos comienzan a divergir el mejor modelo a guardar.

# Capítulo 4

# Desarrollo

*Dos cosas son infinitas: la estupidez humana y el universo; y no estoy seguro de lo segundo.*

Albert Einstein

## 4.1 Introducción

El objetivo fundamental de este TFG es la implementación de un sistema de detección de acciones a partir de secuencias de imágenes de profundidad, tal y como se plantea en el capítulo 1. El esquema general del sistema desarrollado se muestra en la figura 4.1. Esta figura ya se ha mostrado en la introducción, pero se repite en esta sección para facilitar la comprensión y lectura del documento.

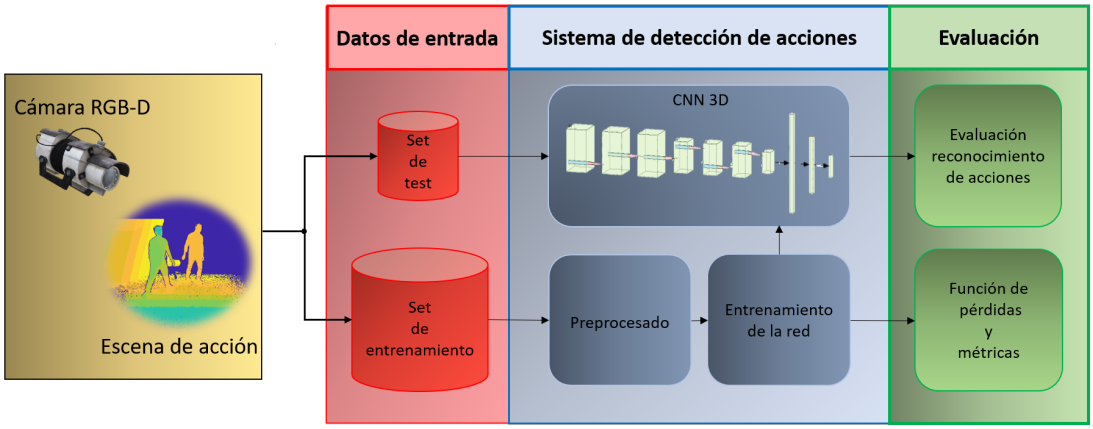



Figura 4.1: Esquema general del sistema propuesto para el reconocimiento de acciones en secuencias de imágenes de profundidad.

El desarrollo del sistema ha consistido en dos etapas fundamentales:

1. **Definición y creación de la arquitectura de red.** Primeramente ha sido necesario el desarrollo de la arquitectura de red, formada por capas de convolución 3D que permitan realizar operaciones sobre secuencias de imágenes, capas de *pooling*, técnicas de regularización...etc.

2. **Entrenamiento de la red.** En este apartado se incluye la explicación del proceso de entrenamiento elaborado, donde ha sido necesario la selección de una función de pérdidas y de un optimizador que

permita minimizarla, además de configurar el tamaño de *batch* y el número de épocas. Asimismo, se expone la creación y división de la base de datos que permite el entrenamiento.

Cada una de las etapas desarrolladas para la consecución del sistema reflejado en la figura 4.1 se describen con detalle a lo largo de este capítulo.

## 4.2 Arquitectura de la red

La elección y definición de la arquitectura de una red neuronal es un elemento crucial en el desempeño del objetivo marcado. Los diferentes tipos de construcciones y arquitecturas presentan ventajas e inconvenientes que pueden marcar el funcionamiento del sistema de reconocimiento de acciones a implementar. La definición de la arquitectura completa de la red se muestra en la figura 4.2. En ella, se indican las dimensiones de cada capa junto con una leyenda de colores que permite observar de forma más clara el tipo de capas que forman la red.

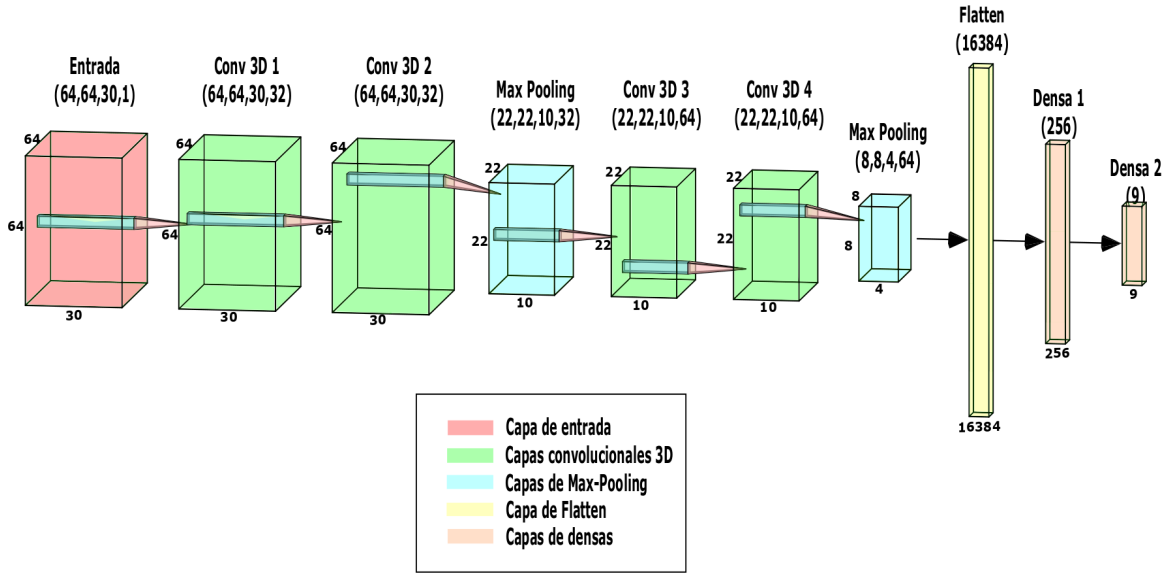



Figura 4.2: Arquitectura de la 3D-CNN implementada.

En la tarea de detección de acciones, se debe trabajar con secuencias de imágenes que contengan toda la definición de la acción considerada. Por ello, la entrada a la red es una secuencia de imágenes de profundidad. Se han considerado imágenes de dimensiones de 64x64 píxeles debido principalmente a la gran cantidad de datos que la red debe procesar y con un número de *frames* estático. En concreto, se trata de un fragmento de vídeo de duración 1 segundo (30 *frames*), mostrando la ejecución de una acción determinada. El empleo de secuencias con un número de *frames* reducido, permite una mayor velocidad en el procesado de los vídeos, sin embargo, puede suceder que la acción no se muestre completa por las diferencias en el tiempo de ejecución de las acciones consideradas. En contraposición, el uso de secuencias más largas incrementa la carga computacional, reduciendo la velocidad en su procesado y pueden aparecer solapamiento entre diferentes acciones. En el desarrollo de este trabajo, se ha realizado un ajuste experimental del número de *frames* para cada secuencia, con el objetivo de alcanzar un equilibrio entre el tiempo de cómputo y la precisión del algoritmo, determinando una longitud de las secuencias de 1 segundo (30 *frames*).

Se debe destacar que ha sido necesario adaptar la red neuronal a los datos de entrada (secuencias de mapas de profundidad) y a las distintas acciones consideradas en este TFG, pues inicialmente la arquitectura propuesta estaba pensada para la detección de diferentes acciones, en concreto, a las pertenecientes a la base de datos *UCF-101* [59] con vídeos RGB con un tamaño de imagen diferente. Por ello, se han modificado las capas de entrada y salida. La primera de ellas, se ha modificado para que acepte secuencias de imágenes con un único canal, el de profundidad, con el objetivo de extraer las características más representativas de las acciones ejecutadas con únicamente la información de profundidad. Por último, la capa densa de salida se ha adaptado con el empleo de 9 neuronas, una por cada clase de acción considerada en este TFG.

La secuencia de entrada se procesa aplicando operaciones de convolución 3D a través de las capas *Conv3D 1* y *Conv3D 2* con 32 filtros cada una de ellas. Posteriormente, se aplica una reducción de dimensionalidad del tensor así como la búsqueda de las mayores activaciones mediante la capa *Max Pooling 1* y se vuelve a introducir en filtros convolucionales mediante las capas *Conv3D 3* y *Conv3D 4* (64 filtros cada una) extrayendo así características con un nivel de abstracción mayor. A continuación, se reduce de nuevo la dimensionalidad y se extrae las características más importantes con la capa *Max Pooling 2* y se introduce el tensor de salida en una capa de *Flatten* que permite la adaptación a las capas finales densas *Dense 1* y *Dense 2* con 256 y 9 neuronas respectivamente. Finalmente, mediante la aplicación de una capa *Softmax* empleada para convertir el vector de salida con valores reales, marcados por la acción de entrada considerada, en valores normalizados en el rango entre [0,1] (ecuación 4.1), se obtiene una probabilidad por cada acción previamente entrenada. La suma del vector de salida resultante debe ser 1, de esa forma sólo existe una clase más probable. En la expresión 4.2 se muestra la ecuación aplicada para la obtención de las probabilidades por cada acción.

$$S\left(a\right) = \begin{bmatrix} a_1 \\ a_2 \\ ... \\ a_9 \end{bmatrix} \rightarrow \begin{bmatrix} S_1 \\ S_2 \\ ... \\ S_9 \end{bmatrix} \tag{4.1}$$

$$S_j = \frac{e^{a_j}}{\sum_{k=1}^{9} e^{a_k}} \qquad j \in 1.,9 \tag{4.2}$$

Un ejemplo práctico del funcionamiento de la capa *SoftMax* se muestra en la figura 4.3. En ella se observa cómo los valores del vector de entrada más altos, se presentan con una probabilidad mayor, mientras que los más bajos les corresponde una probabilidad menor. De esta forma, en el contexto del sistema de detección de acciones implementado, la capa final de densas proporciona un vector de 9 elementos, uno por acción, cuyo elemento con mayor activación será el representado por una mayor probabilidad y escogido como candidato para proporcionar una detección de acción de salida.


Figura 4.3: Ejemplo del funcionamiento de la capa SoftMax.

Tanto las capas convolucionales 3D como las de *Pooling*, se realizan con un parche de dimensiones (3,3,3). Las dimensiones de las imágenes (64,64,30), no se modifican al aplicar los filtros convolucionales ya que se emplea el *padding* para evitar la reducción de dimensionalidad. A la salida de cada capa, se incluye una función de activación de tipo *ReLu* por las ventajas que se mencionaron en el capítulo 3 entre las que destacan su alta eficiencia y la capacidad de proporcionar la no linealidad necesaria para la resolución del reconocimiento de acciones. Se debe destacar el empleo de la técnica de regularización de *Dropout* que consiste en ignorar nodos de manera aleatoria con el objetivo de evitar las interdependencias entre ellos, es decir, los nodos no aprenden funciones que se basan en valores de entrada de otro nodo, por tanto, el *Dropout* permite a la red aprender una relación más sólida. Es necesario recordar que el *Dropout* sólo actúa en la etapa de entrenamiento entre la primera capa de *Max Pooling* y la capa *Conv3D 3* y entre la segunda de *Max Pooling* y la capa de *Flatten*.

En la Tabla 4.1 se muestra un resumen de las diferentes capas que conforman la red, así como los tamaños de salida y los parámetros fundamentales.

| **Capas red neuronal convolucional 3D** | | |
|---|---|---|
| **Capa** | **Tamaño de salida** | **Parámetros** |
| Entrada | $64 \times 64 \times 30 \times 1$ | - |
| Conv3D 1 | $64 \times 64 \times 30 \times 32$ | kernel=(3, 3, 3) / strides=(1, 1, 1) |
| Activación | ReLU | |
| Conv3D 2 | $64 \times 64 \times 30 \times 32$ | kernel=(3, 3, 3) / strides=(1, 1, 1) |
| Activación | ReLU | |
| MaxPooling | $22 \times 22 \times 10 \times 32$ | size=(3, 3, 3) |
| Dropout | 0.25 | |
| Conv3D 3 | $22 \times 22 \times 10 \times 64$ | kernel=(3, 3, 3) / strides=(1, 1, 1) |
| Activación | ReLU | |
| Conv3D 4 | $22 \times 22 \times 10 \times 64$ | kernel=(3, 3, 3) / strides=(1, 1, 1) |
| Activación | ReLU | |
| MaxPooling | $8 \times 8 \times 4 \times 64$ | size=(3, 3, 3) |
| Dropout | 0.25 | |
| Flatten | 16384 | - |
| Densa 1 | 256 | - |
| Activación | ReLU | |
| Dropout | 0.5 | |
| Densa 2 | 9 | - |
| Activación | SoftMax | |

Tabla 4.1: Arquitectura de la red y tamaño de los tensores de cada capa.

## 4.3 Entrenamiento

En esta sección se comentan los aspectos más relevantes que permiten el entrenamiento de la red expuesta en el apartado anterior. Además, se incluye una descripción de la creación y división de la base de datos empleada. A continuación, se enumeran la elección de los parámetros más importantes para un correcto entrenamiento:

1. **Optimizador.** El optimizador elegido para este trabajo es el optimizador *Adam* debido a sus capacidades adaptativas. Dado que el tamaño de batch seleccionado, es lo suficientemente grande, permite un correcto funcionamiento del optimizador al mejorar la convergencia de los métodos adaptativos. Además, se debe mencionar que *Adam* ha demostrado que realiza entrenamiento más

rápidos y más estables, es decir, sin introducir sobreimpulsos en la función de pérdidas comparado con [60].

2. **Función de pérdidas.** Puesto que el objetivo del sistema implementado está centrado en el reconocimiento de acciones, es decir, es un problema de clasificación multi-clase, se ha seleccionado la función de pérdidas CCE ya que proporciona una convergencia más rápida y es más común lograr una optimización a un mínimo global.

3. **Tamaño de batch.** El tamaño de batch empleado es de 32 secuencias de 30 imágenes por batch. Sin embargo, se han realizado diferentes entrenamientos con tamaños que oscilan entre las 8 y las 64 secuencias de imágenes por batch. En el capítulo 5 se exponen las consecuencias de la modificación en el tamaño de batch en la precisión obtenida.

4. **Número de épocas.** El número de épocas empleado es de 50, pero los pesos que se almacenan son los que consiguen una mayor tasa de precisión en validación durante todo el proceso de entrenamiento. Esto es debido a que se emplea una función intra-entrenamiento que permite guardar los mejores pesos resultantes durante el mismo.

Una cuestión fundamental es el modo de selección de los 30 *frames* que forman una determinada acción. Puesto que cada una de ellas tiene un tiempo de ejecución distinto, ha sido necesario escoger 30 imágenes que cubrieran el contexto general de la acción, es decir, en función de la longitud total del vídeo en cuestión, se obtiene un valor de salto de *frame* que permite determinar cuáles escoger. En el supuesto de trabajar con una secuencia de vídeo cuyo número de *frames* es inferior a 30, se ha realizado un procesado para forzar la repetición de dicha acción hasta conseguir el número de *frames* mínimos necesarios.

Un ejemplo de lo anterior, se muestra en la figura 4.4. En ella, se observa una secuencia de imágenes de una determinada acción formada por 150 *frames*, obteniendo un salto de 5 *frames* que permite seleccionar aquellos 30 *frames* que cubren todo el contexto de la acción.

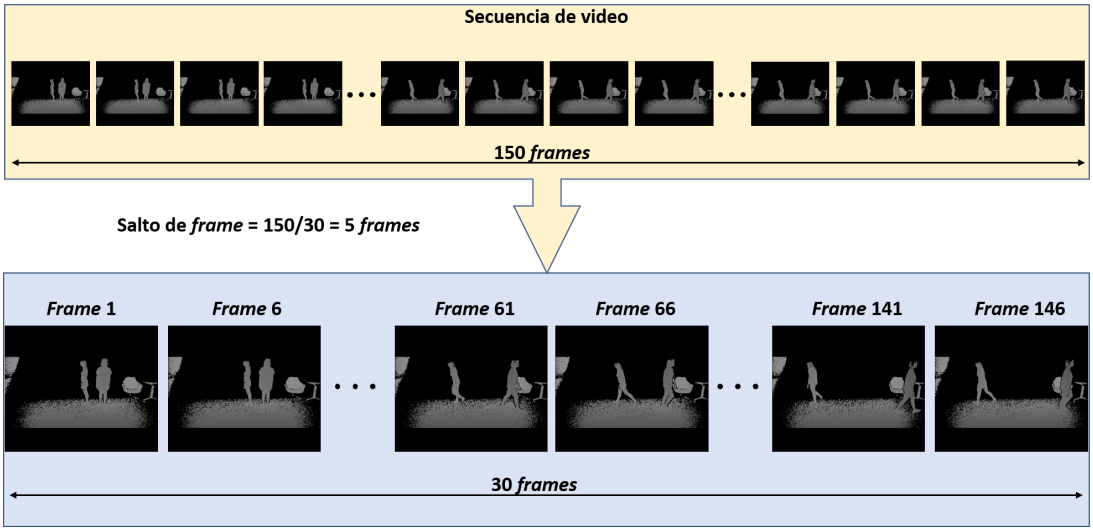



Figura 4.4: Ejemplo de selección de 30 *frames* en un video.

El proceso de entrenamiento consiste en introducir a la red un conjunto de entradas y sus correspondientes salidas deseadas. En este caso, las entradas son secuencias de imágenes de 64x64 píxeles, que permiten la perfecta adaptación a la capa de entrada definida en la arquitectura de la red, y las salidas son un vector de 9 elementos, uno por cada clase de acción, de tipo *one hot* [61], es decir, el elemento

correspondiente a la acción de entrada toma el valor 1 y los demás el valor 0. Esta etapa se ha realizado de dos formas bien diferenciadas. En la primera, los datos destinados al entrenamiento y la validación se agrupan en un único bloque, mientras que en la segunda alternativa, se van pasando los datos a la red *batch* a *batch* gracias a un generador, sin formar un único bloque completo a la red. A continuación se describen con mayor detalle las dos propuestas implementadas.

### 4.3.1 Bloque único

En esta primera alternativa, durante la preparación de los datos de entrenamiento, éstos se agrupan en un único bloque que contiene las entradas (secuencias de 30 imágenes de 64x64) y sus salidas correspondientes (vector de 9 elementos de tipo *one hot*. Una vez creado el bloque completo, se introduce a la red neuronal en bloque del tamaño de *batch* seleccionado, en este caso 32. En la figura 4.5 se muestra un ejemplo del proceso de este tipo de entrenamiento.

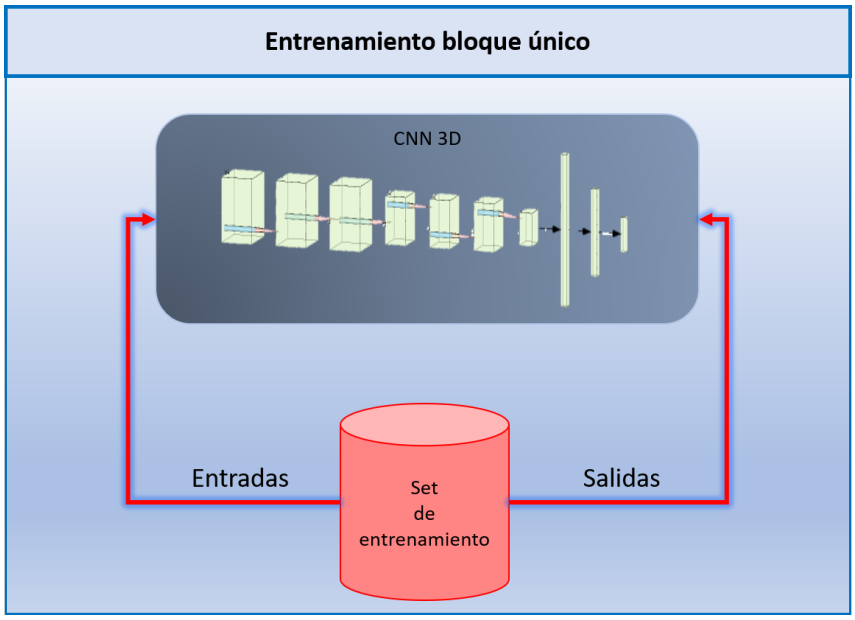



Figura 4.5: Entrenamiento de la 3D-CNN basado en la creación de un bloque único.

Una de las principales ventajas que presenta esta alternativa es su simplicidad y facilidad para la realización del entrenamiento, pero tiene una serie de inconvenientes relacionados con la aleatoriedad de los datos, ya que una vez creado el bloque de entrenamiento, la secuencia de *batches* que se introducen a la red a través de todas las épocas es la misma, perjudicando en su correcta generalización. Además, se debe destacar que presenta problemas de memoria trabajando con bases de datos grandes, pues el bloque completo debe estar presente en ella. Por todo lo anterior, se ha optado por el empleo de la segunda alternativa, basada en un generador de entrenamiento.

### 4.3.2 Generador de entrenamiento

Esta segunda alternativa se basa en el empleo de una función generador. El generador permite la preparación de los datos de entrenamiento seleccionando de manera aleatoria las secuencias de imágenes necesarias para la creación de los *batches* de manera concurrente y en paralelo al proceso de entrenamiento de la red neuronal. En la figura 4.6 se muestra un ejemplo del proceso de entrenamiento utilizando una función generador.

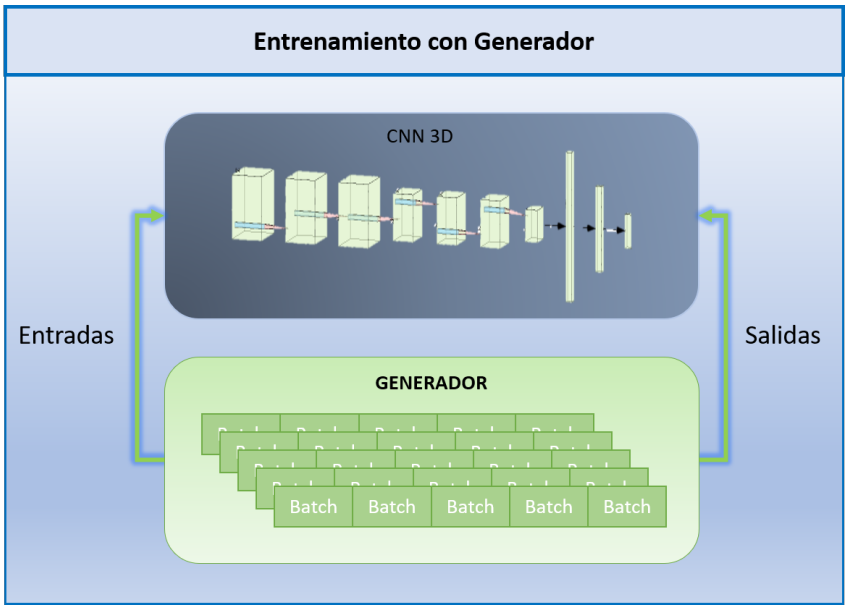



Figura 4.6: Entrenamiento de la 3D-CNN basado en el empleo de un generador.

El generador permite evitar los problemas de memoria que suceden en la alternativa comentada en 4.3.1, ya que únicamente permanecen en memoria en cada momento los datos de entrada y salida pertenecientes a un *batch*, además, se asegura una mayor aleatoriedad en la selección de los vídeos destinados al entrenamiento, siendo posible un aumento de datos *online*.

## 4.4 Preparación de la base de datos

Como ya se ha comentado en el capítulo 1, la base de datos empleada para el entrenamiento de la 3D-CNN es la *NTU* [13, 14] tanto para el entrenamiento como para la evaluación de la red, ya que proporciona una gran cantidad de vídeos de acciones con información de profundidad.

La base de datos *"NTU RGB+D Action Recognition"* está compuesta por 56.880 muestras de vídeo de una o varias personas ejecutando una determinada acción. Por cada secuencia se incluyen: vídeos RGB, mapas de profundidad, esqueletos en 3D e imagen de infrarrojos. Un ejemplo de la información disponible por la base de datos se muestra en la figura 4.7. En ella, se observan imágenes de distinto tipo representando un *apretón de manos*.

Figura 4.7: Ejemplo de la información proporcionada por la base de datos. Se incluye información RGB, RGB+*joints*, *depth*, *depth*+*joints* e información infrarroja [13, 14].

Se han obtenido a partir de la grabación simultánea de 3 cámaras *Microsoft Kinect v.2* para proveer a la base de datos de diferentes puntos de vista. La resolución de los vídeos RGB es de 1920x1080 píxeles, mientras que los mapas de profundidad y los vídeos de infrarrojos tienen una resolución de 512x424 píxeles. Por otra parte, los datos de esqueletos en 3D contienen las localizaciones en 3 dimensiones de las

principales 25 articulaciones del cuerpo para cada *frame*. La base de datos contiene 60 clases de acciones que se pueden agrupar en 3 grandes grupos: acciones diarias, acciones mutuas y condiciones médicas. En el primer grupo se incluyen acciones como *beber agua*, *lavarse los dientes* o *jugar con el móvil*. En cuanto a las acciones mutuas se pueden mencionar *darse un apretón de manos*, *empujarse* o *abrazarse*. Por último, en condiciones médicas se pueden citar *toser*, *caerse* o *nauseas*. En el presente trabajo se han seleccionado 9 acciones, que se pueden dar en el contexto de la vídeo-vigilancia y la seguridad. Las acciones escogidas junto con el número de fragmentos empleados tanto para el entrenamiento y validación como para el test se muestran en la tabla 4.2. De toda la información que proporciona la base de datos *NTU*, se debe recordar que en este TFG solo se emplean los mapas de profundidad. Los valores de profundidad están codificados en 16 bits y representan la distancia de cada punto a la escena en milímetros. El rango máximo de valores en los mapas de profundidad de las acciones seleccionadas se encuentra entre 0 y 8000.

En total, se han empleado 8532 secuencias de mapas de profundidad, con distintos individuos realizando las acciones descritas en la tabla 1.1, desde diferentes puntos de vista.

| **Acciones utilizadas de la base de datos NTU** | | | | |
|---|---|---|---|---|
| **Acción** | **Cód.** | **Entren.** | **Validación** | **Test** |
| Propinar un puñetazo | A01 | 607 | 151 | 190 |
| Levantarse | A02 | 607 | 151 | 190 |
| Caerse | A03 | 607 | 151 | 190 |
| Sentarse | A04 | 607 | 151 | 190 |
| Separarse de una persona | A05 | 607 | 151 | 190 |
| Propinar una patada | A06 | 607 | 151 | 190 |
| Empujar | A07 | 607 | 151 | 190 |
| Andar hacia una persona | A08 | 607 | 151 | 190 |
| Lanzar un objeto | A09 | 607 | 151 | 190 |
| Total | - | 5463 | 1359 | 1710 |

Tabla 4.2: Datos utilizados en la etapa de entrenamiento de la red.

De las secuencias correspondientes a las 9 acciones seleccionadas, se ha utilizado el 80 % para entrenamiento (6822), que a su vez de han dividido de forma que: el 80 % se emplea para el entrenamiento y el 20 % para validar. El 20 % restante del total (1710) se emplea para la etapa de test y la extracción de resultados. El empleo de muestras con una alta variabilidad, es decir, un número considerable de individuos realizando acciones desde diferentes puntos de vista, permite aumentar el nivel de generalización de la red y, por tanto, la robustez del sistema.

Se debe asegurar la normalización de las secuencias de imágenes de profundidad que se introducen a la red. De esta forma, el aprendizaje desarrollado durante la etapa de entrenamiento permite aumentar la robustez del sistema ante secuencias de vídeos de acciones de bases de datos distintas.

La normalización se realiza mediante la búsqueda del máximo valor de píxel en las secuencias de imágenes de acciones pertenecientes a la base de datos. Tras lo anterior, se dividen todas las imágenes entre el valor máximo encontrado, de esta forma, se consigue tener la misma información de profundidad representada en el rango entre 0 y 1. En la figura 4.8 se muestra un ejemplo de normalización de una imagen.

### 4.4.1 Técnicas de mejora del entrenamiento

Con el objetivo fundamental de obtener un valor de precisión mayor y aumentar la generalización de la red, se describen a continuación una serie de técnicas de aumento de datos empleadas para cumplir el objetivo marcado.

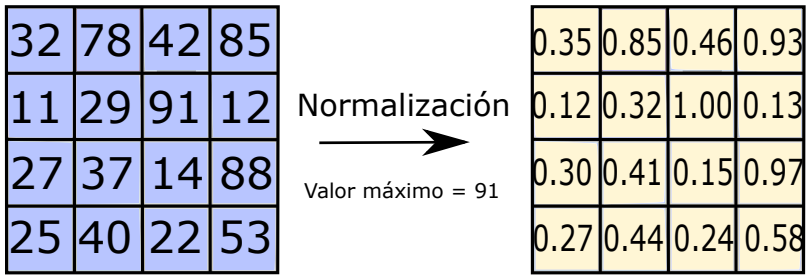



Figura 4.8: Ejemplo de normalización de una imagen 4x4 píxeles.

- **Adicción de ruido.** La aplicación de ruido gaussiano sobre las secuencias de imágenes de acción, introduce una componente adicional de aleatoriedad que beneficia a la red en el proceso de generalización de los datos. Un ejemplo del ruido aplicado en los *frames* que forman los vídeos de acción se muestra en la figura 4.9.

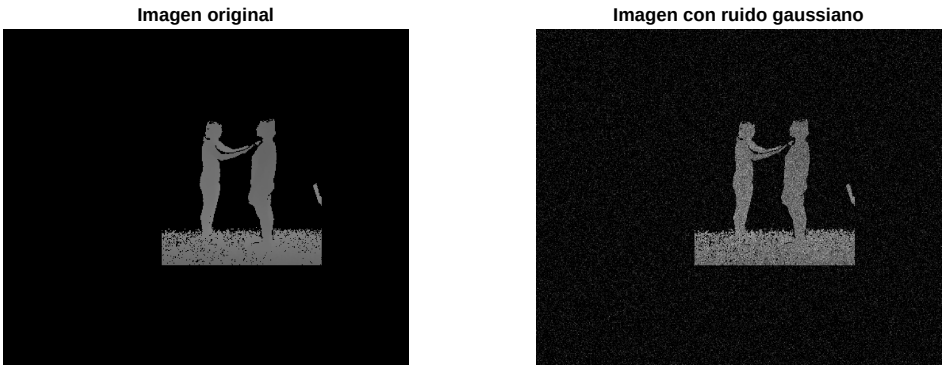



Figura 4.9: Ejemplo de adicción de ruido sobre las muestras de entrenamiento.

- **Espejo.** La aplicación de la técnica de espejo o *mirror* permite aumentar considerablemente las muestras de entrenamiento y su variabilidad, produciendo secuencias de vídeo con distintas perspectivas. En la figura 4.10 se muestra un ejemplo de aplicación de la técnica de espejo sobre una imagen de profundidad.

- **Rotaciones.** El empleo de pequeñas rotaciones sobre las secuencias de imágenes de profundidad permite el aumento de muestras de datos para el entrenamiento. Esto posibilita simular pequeñas rotaciones en la cámara en la adquisición de las imágenes. Las posibles rotaciones oscilan entre los -5 y 5 grados. En la figura 4.11 se muestra un ejemplo de aplicación de una rotación de 4º.

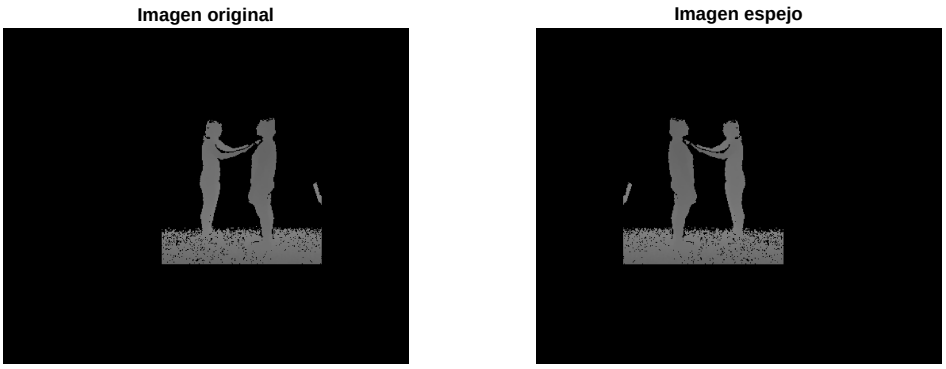



Figura 4.10: Ejemplo de aplicación de la técnica de espejo.

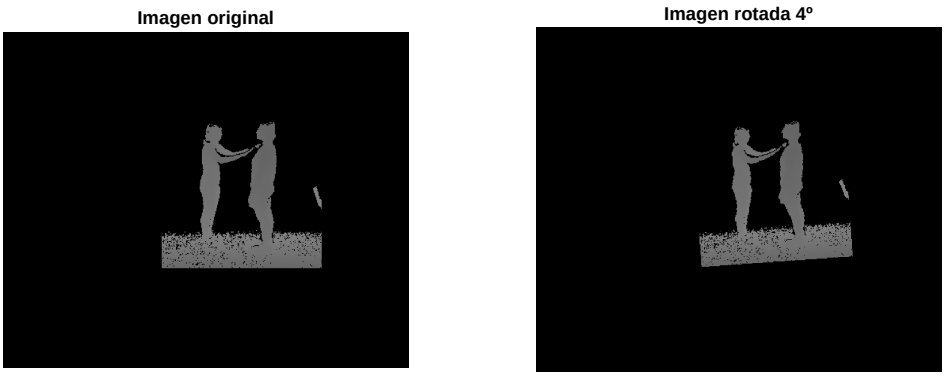



Figura 4.11: Ejemplo de aplicación de rotación sobre la imagen de profundidad.

# Capítulo 5

# Resultados

*Simplicidad. Ésta es la más innovadora sofisticación.*

Leonardo Da Vinci

## 5.1 Introducción

En este capítulo se describen los resultados obtenidos mediante el sistema de detección de acciones implementado en este TFG. Se debe mencionar que todos los resultados que se presentan se han obtenido empleando las bases de datos que se describen en 5.1.1.

Primeramente se detallan las bases de datos empleadas y su origen. Posteriormente se realiza una explicación de las métricas empleadas para medir el desempeño del sistema desarrollado. Por último, se incluyen los resultados experimentales sobre cada una de las bases de datos consideradas.

### 5.1.1 Bases de datos utilizadas

A continuación se describen las bases de datos empleadas para la consecución marcada en este TFG. Ambas están centradas en el reconocimiento de acciones, y son ampliamente utilizadas para la evaluación de dicho propósito.

#### 5.1.1.1 NTU

Como se ha expuesto en el apartado 4.4, la base de datos *NTU RGB+D Action Recognition* está compuesta por 56.880 muestras de vídeo de una o varias personas ejecutando una determinada acción. Por cada secuencia se incluyen: vídeos RGB, mapas de profundidad, esqueletos en 3D e imagen de infrarrojos. Se han obtenido a partir de la grabación simultánea de 3 cámaras *Microsoft Kinect v.2* para proveer a la base de datos de diferentes puntos de vista. La resolución de los vídeos RGB es de 1920x1080 píxeles, mientras que los mapas de profundidad y los vídeos de infrarrojos son tienen una resolución de 512x424 píxeles. Por otra parte, los datos de esqueletos en 3D contienen las localizaciones en 3 dimensiones de las 25 principales articulaciones del cuerpo para cada *frame*. La base de datos contiene 60 clases de acciones que se pueden agrupar en 3 grandes grupos: acciones diarias, acciones mutuas y condiciones médicas. En el primer grupo se incluyen acciones como *beber agua*, *lavarse los dientes* o *jugar con el móvil*. En cuanto a las acciones mutuas se pueden mencionar *darse un apretón de manos*, *empujarse* o *abrazarse*. Por último, en condiciones médicas se pueden citar *toser*, *caerse* o *nauseas*.

En total, se han empleado 8532 secuencias de mapas de profundidad, con distintos individuos realizando las acciones descritas en la tabla 1.1, desde diferentes puntos de vista. En la figura 5.1 se muestran algunos ejemplos de las imágenes de profundidad utilizadas. En esta figura se ha utilizado una escala de color para representarlos diferentes valores de profundidad medidos. Además, como se puede observar, en la información proporcionada se ha eliminado el fondo de las escenas.

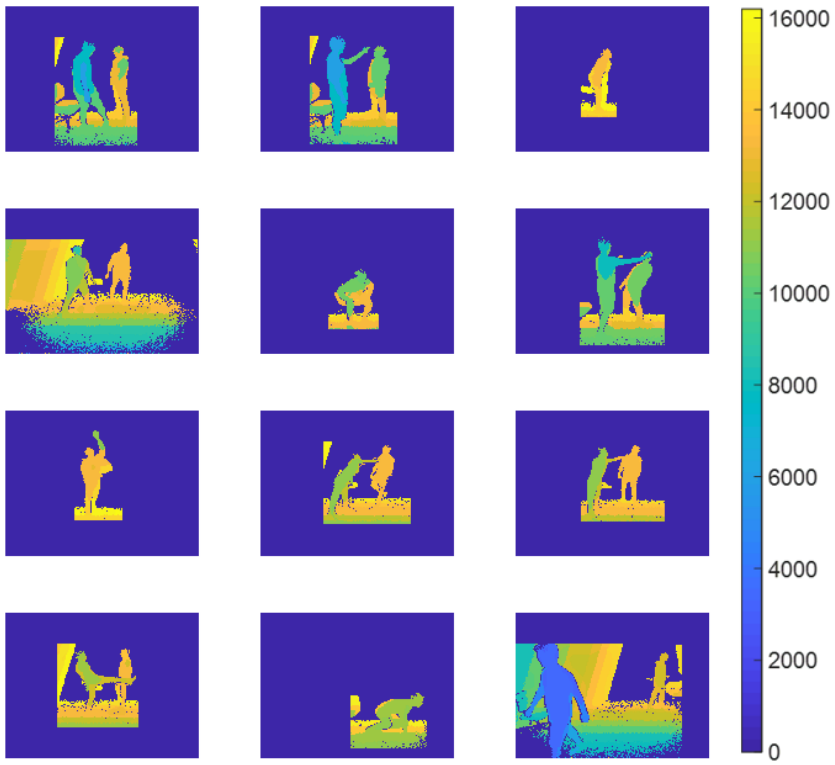



Figura 5.1: Ejemplos de escenas de profundidad pertenecientes a diferentes secuencias de vídeo de la base de datos NTU, con varios individuos realizando una serie de acciones desde diferentes puntos de vista.

Dado el gran número de muestras pertenecientes a esta base de datos, es la empleada para realizar el entrenamiento de la red neuronal y, posteriormente la validación del sistema con los vídeos destinados a ese fin.

#### 5.1.1.2 UTKinect

La base de datos *UTKinect-Action3D* [32] fue desarrollada como parte de un trabajo de investigación en reconocimiento de acciones. Está compuesta por vídeos capturados por una única cámara fija *Kinect v1*. En total se dispone de 10 tipos de acciones: *andar, sentarse, levantarse, lanzar un objeto, llevar un objeto, empujar, tirar, saludar y dar palmas*. Hay 10 sujetos que realizan cada acción, 2 veces cada uno, situados desde diferentes puntos de vista. Por cada secuencia se incluyen: vídeos RGB, mapas de profundidad y esqueletos en 3D. La tasa de imagen o *framerate* es de 30 *frames* por segundo. La resolución de las secuencias de imágenes RGB es de 480x620 píxeles, mientras que los mapas de profundidad tienen una resolución de 320x240 píxeles.

Del conjunto de acciones que contiene la base de datos, únicamente interesan las que sean comunes a la base de datos NTU (la empleada para el entrenamiento) entre las que se incluyen *andar, sentarse, levantarse y empujar*. En la figura 5.2 se muestran algunos ejemplos de las imágenes pertenecientes a esta base de datos.

Figura 5.2: Ejemplos de escenas RGB y de profundidad pertenecientes a diferentes secuencias de vídeo de la base de datos UTKinect, con varios individuos realizando una serie de acciones.

Esta base de datos se emplea como comprobación de la generalización realizada por el sistema tras el entrenamiento, ya que comparte varias acciones en común y presenta un punto de vista y escenario distinto al empleado para entrenar.

### 5.1.2 Métricas de calidad

En esta sección se definen las métricas de calidad empleadas para la evaluación del sistema de detección de acciones implementado en este TFG.

En este caso, la única métrica empleada para determinar la precisión en la clasificación de acciones se basa en la comparación entre el vector de salida etiquetado y el vector de salida predicho. Este último proporciona un valor entre 0 y 1 por cada acción considerada en el vector de salida, siendo un 1 el valor más alto en la predicción por parte de la red neuronal y un 0 el valor más bajo.

La obtención de la precisión del sistema de reconocimiento de acciones se obtiene mediante la comparación de la posición del valor "1"de los vectores de salida etiquetados en formato *one hot* y la posición con mayor probabilidad de los vectores de salida de la red neuronal, siempre y cuando dicha probabilidad sea superior al umbral de 0.5 (50 %). Se ha considerado un umbral del 50 % para tener únicamente en consideración a las acciones que detecta la red con mayor certidumbre. El procedimiento descrito anteriormente, se resumen en el algoritmo 5.1. Una vez obtenida la precisión se genera una matriz de confusión que permita visualizar de forma gráfica el comportamiento del sistema.

Una matriz de confusión es una representación de los datos gráfica, ampliamente utilizada como métrica de calidad en problemas de clasificación. En ella, se presenta en cada fila las clases reales y en las columnas las correspondientes clases predichas. De esta forma, se puede interpretar de forma sencilla la confusión del sistema entre clases. Un ejemplo de matriz de confusión se muestra en la figura 5.3. En ella, se presenta un problema de clasificación con 3 clases: gatos, perros y conejos con un total de 27 muestras, representando la densidad de los resultados por cada clase en escala de grises, empleando el color negro como la mayor densidad de datos y el blanco como la menor. Se puede observar como el sistema de clasificación de ejemplo propuesto tiene dificultades en distinguir entre las clases de perros y gatos, pero es capaz de realizar una buena distinción entre conejos y otro tipo de animales.

```
Inicialización: Vector de salida real;
Vector de salida predicha;
Aciertos = 0;
for Número de vídeos de test do
    PosVectorReal = MaxPos(Vector de salida real);
    PosVectorRed = MaxPos(Vector de salida predicha);
    if PosVectorReal == PosVectorRed then
        if MaxValor(Vector de salida predicha) ≥ 0.5 then
            Aciertos+=1;
        end
    end
end
Precisión = Aciertos/Número de vídeos de test;
```

**Algoritmo 5.1:** Obtención de precisión categórica

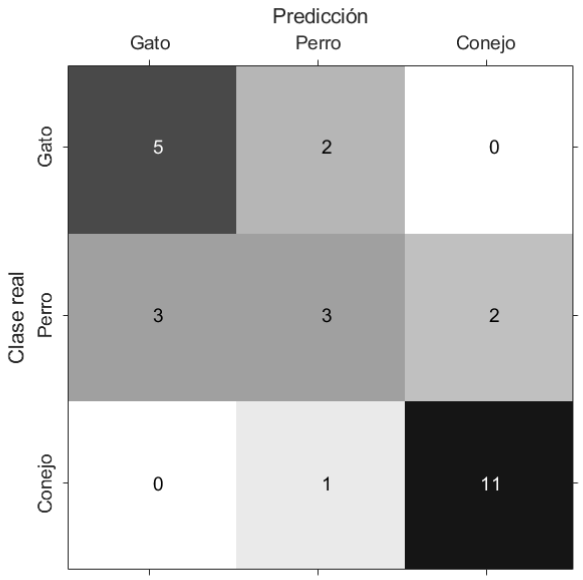



Figura 5.3: Ejemplo de matriz de confusión en un problema de clasificación de animales.

El formato de matriz de confusión mostrado en la figura 5.3 se ha empleado para la visualización de los resultados obtenidos en la evaluación del sistema de detección de acciones desarrollado en este TFG.

### 5.1.3 Estrategia y metodología de experimentación

A continuación se procede a comentar la estrategia de evaluación que se ha seguido. Este punto es fundamental, pues marca la calidad de la evaluación realizada y la robustez del sistema.

Primeramente se han realizado las etapas de entrenamiento y evaluación con los datos de la base de datos *NTU* en un único bloque, tal y como se describe en el apartado 4.3.1. Posteriormente, se ha efectuado lo anterior con las mismas muestras de entrenamiento junto con una función generador, tal y como se describe en el apartado 4.3.2. Tras lo anterior y a la vista de los resultados que se muestran en la tabla 5.1 en el punto 5.2, se ha decidido proseguir la experimentación y evaluación del sistema únicamente con la propuesta del generador. Una vez elegida la propuesta, se ha realizado un estudio experimental del tamaño de *batch* más adecuado para el problema de la detección de acciones planteado. Por último, la evaluación de la red se ha efectuado con las bases de datos descritas en 5.1.1, a través de la generación de un fichero de resultados que permita la comparación entre los vectores de salida etiquetados y predichos.

La obtención y comparación de las precisiones obtenidas del sistema de reconocimiento de acciones se ha realizado tal y como se ha descrito en el apartado 5.1.2.

## 5.2 Resultados experimentales

En este apartado se presentan todos los resultados experimentales obtenidos. Se comenzará por la presentación de los resultados obtenidos por las dos alternativas de entrenamiento descritas. Posteriormente, se muestra una gráfica que presenta el comportamiento de la red en función del tamaño de *batch* seleccionado junto con los resultados de precisión y de función de pérdidas durante el proceso de entrenamiento. Finalmente, se exponen las matrices de confusión obtenidas para cada una de las bases de datos consideradas. Todo ello, viene acompañado de comentarios al respecto que permitan la extracción de conclusiones.

### 5.2.1 Comparación alternativas de entrenamiento

En la tabla 5.1 se muestra la comparación entre las precisiones obtenidas utilizando las diferentes alternativas. En ella, se observan las ventajas que presenta el empleo de una función generador que garantice una suficiente aleatoriedad de los datos, que se traduce en una mayor generalización por parte de la red desarrollada.

| **Comparación de precisión** | |
|---|---|
| **Alternativa** | **Precisión ( %)** |
| Bloque único | 86.91 % |
| Generador | 94.13 % |

Tabla 5.1: Comparación entre las alternativas de entrenamiento: Bloque único y Generador.

Además, se debe mencionar que el empleo de un generador permite eludir los problemas relativos a la memoria que presenta la alternativa de bloque único. Lo anterior, ha permitido la mejora del entrenamiento de la red a través del aumento de datos.

### 5.2.2 Resultados del entrenamiento

En la figura 5.4 se muestra de forma gráfica el estudio experimental realizado para la selección de un tamaño de batch que maximice la precisión en la clasificación de acciones en la base de datos considerada para el entrenamiento.

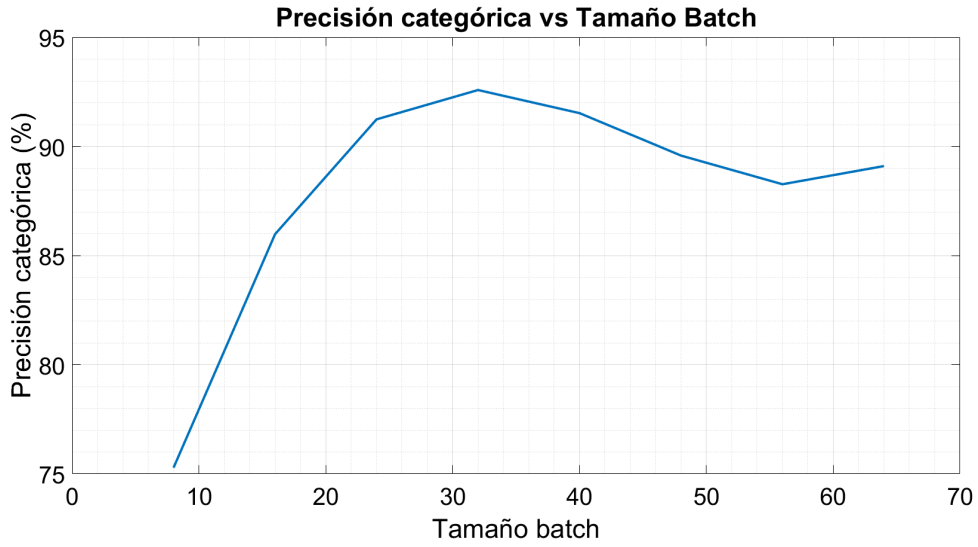


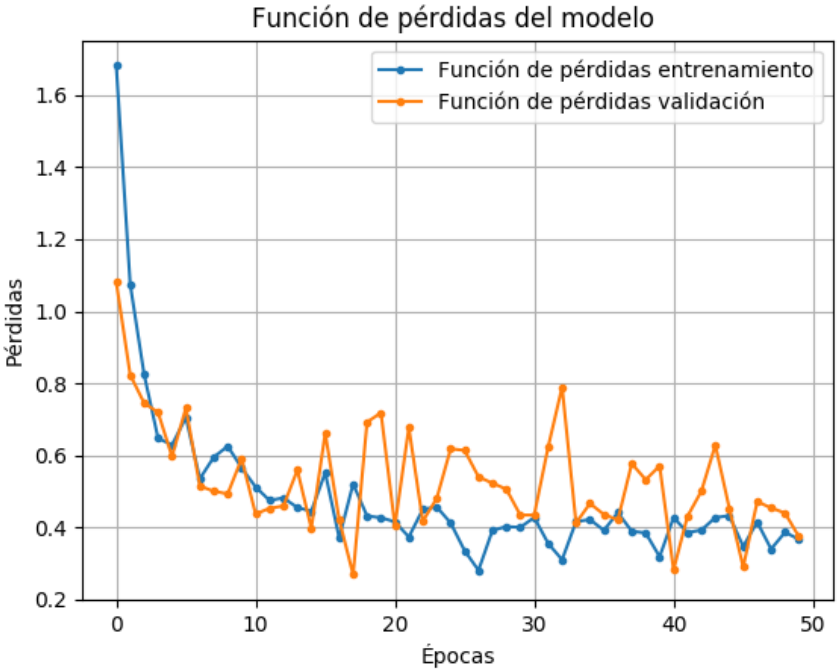

Figura 5.4: Precisión obtenida durante el entrenamiento en función del tamaño de batch seleccionado.

A la vista de los resultados obtenidos, se ha empleado un tamaño de batch de 32 secuencias de imágenes por batch. En las figura 5.5 se muestra el valor de la función de pérdidas 5.5a y la precisión obtenida 5.5b durante las 50 épocas consideradas en la etapa de entrenamiento con un tamaño de batch de 32. Se puede observar en las curvas de entrenamiento como la 3D-CNN implementada alcanza el régimen oscilatorio final sin mostrar rasgos de sobreentrenamiento dado que la precisión categórica de validación tiene un valor cercano a la de entrenamiento, lo cual permite deducir que se ha realizado una correcta generalización del entrenamiento por parte de la red neuronal. Además, el algoritmo de optimización *Adam* permite evitar el problema de los sobre-impulsos en la minimización de la función de pérdidas, tal y como se expuso en 3.3.4.

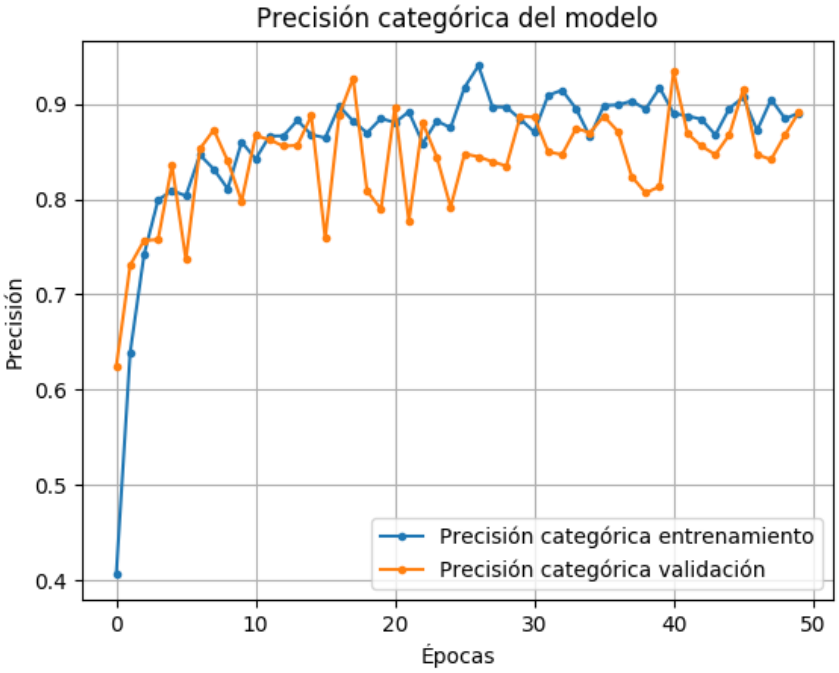
### 5.2.3 Evaluación de la red

En este apartado se muestran los resultados obtenidos fruto de la evaluación de la red implementada con las bases de datos descritas en 5.1.1. Primeramente, en la figura 5.6 se muestra la matriz de confusión obtenida a través de las secuencias de video de acciones destinadas para test. En esta figura, se representan las acciones por su código de acción definido en la Tabla 1.1. En las filas de la matriz se representa la clase real, es decir, los vídeos de acción que se encuentran etiquetados, y en las columnas, se representa la clase predicha, la predicción que proporciona la red neuronal como salida ante un determinado vídeo de entrada. Se debe mencionar que se ha incluido una clase ficticia denominada *none* que permita visualizar los casos en los que la red proporciona unos valores de probabilidad por acción muy bajos, es decir, ninguna clase predicha supera el umbral mínimo marcado (50 %) y, por tanto, dicha secuencia se incluye en la clase ficticia. Es por ello que en la última fila donde se representa la clase *none* aparece el valor *NaN* pues no existen dichos vídeos de acción.

Analizando la matriz de confusión, se observa que la mayor parte de las clases presentan precisiones elevadas (por encima del 90 %). Se deduce por tanto que la red neuronal ha realizado una correcta




(a) Función de pérdidas en cada época durante el entrenamiento.




(b) Precisión categórica obtenida en cada época durante el entrenamiento.

Figura 5.5: Resultados obtenidos durante el entrenamiento. Se incluyen los valores de función de pérdidas 5.5a y la precisión categórica obtenida 5.5b.

generalización en la clasificación de acciones. Las acciones más problemáticas son las de *"propinar un puñetazo"* (A01), *"empujar"* (A07) y la de *"lanzar un objeto"*, entre las que existe cierta confusión, debido principalmente a la dificultad que presenta la base de datos utilizada, al incluir cambios importantes de punto de vista. Así como a la complejidad de la tarea, al utilizar únicamente información de profundidad. Sin embargo, a pesar de este hecho, se consigue un valor de precisión global en la clasificación elevado (94.13 %).

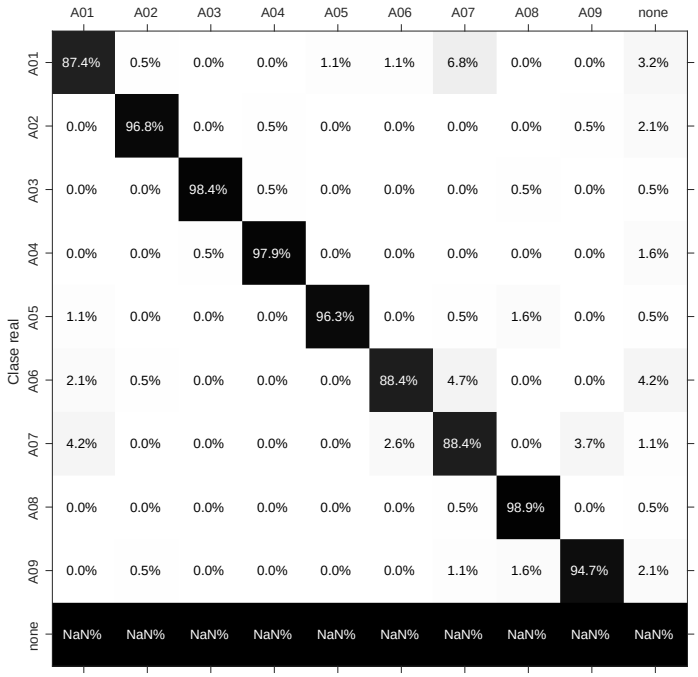



Figura 5.6: Matriz de confusión del sistema de clasificación de acciones realizando el test con la base de datos *NTU*.

En la figura 5.7 se pueden apreciar los resultados obtenidos tras la realización del test con la base de datos *UTKinect*. En la figura se observan las 4 acciones consideradas, comunes tanto para la *NTU* como para la *UTKinect*, con el objetivo de observar el comportamiento de la red ante vídeos de acciones pertenecientes a una base de datos diferente a la empleada para el entrenamiento.

Tal y como se esperaba, la precisión categórica en la clasificación disminuye hasta un valor de 67 %. Esto es debido, a que las secuencias de vídeo presentan un escenario y puntos de vista diferentes a los empleados para el entrenamiento de la red. Sin embargo, se trata de un buen resultado, a pesar del valor de precisión obtenido, pues la red ha sido capaz de clasificar las acciones de una base de datos, cuyas muestras no han sido previamente vistas.

## 5.3 Conclusiones

Tras examinar los resultados previamente expuestos se puede observar el buen comportamiento de las 3D-CNN empleando únicamente información de profundidad para el propósito de la clasificación de acciones en un contexto de la seguridad y la vídeo-vigilancia. A continuación, se presentan los puntos más significativos extraídos de los resultados experimentales realizados:

1. **Alternativas de entrenamiento:** tras los resultados experimentales obtenidos con la alternativa de agrupar todo el set de entrenamiento en un bloque único y la propuesta con una función

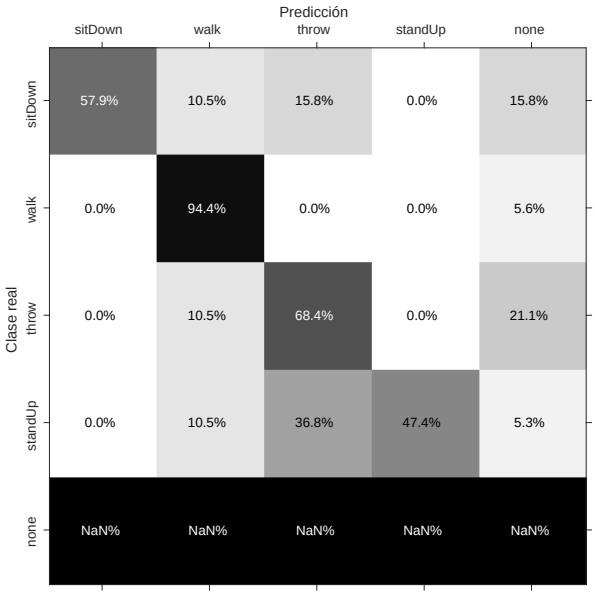



Figura 5.7: Matriz de confusión del sistema de clasificación de acciones realizando el test con la base de datos *UTK*.

generador, se puede observar la mejora introducida por esta última en la precisión categórica de la red. Esto se debe a la mayor aleatoriedad y variabilidad en la preparación de las secuencias de entrenamiento y validación de la red. Además, se debe añadir que el generador utiliza una pequeña parte de la memoria, permitiendo así el aumento de datos para el entrenamiento, sin perjudicar el rendimiento del PC donde se aloje el sistema de detección de acciones.

2. **Tamaño de *batch*:** las pruebas experimentales ejecutadas para determinar el tamaño de *batch* óptimo para la tarea encomendada ha permitido aumentar las precisiones categóricas alcanzables durante el entrenamiento.

3. **Bases de datos:** el empleo de 2 bases de datos diferentes que contienen acciones en común ha permitido la observación del correcto funcionamiento del sistema de reconocimiento en diferentes circunstancias, como son la modificación de escenario y la ejecución de acciones desde diferentes puntos de vista.

Los resultados experimentales obtenidos con las dos bases de datos han permitido la validación del sistema desarrollado para la detección de acciones mediante redes neuronales convolucionales.

# Capítulo 6

# Conclusiones y líneas futuras

En este apartado se resumen las principales conclusiones obtenidas y se proponen algunas posibles futuras líneas de investigación que se deriven del TFG desarrollado.

## 6.1 Conclusiones

En este trabajo se ha presentado un sistema de reconocimiento de acciones empleando únicamente información de profundidad. La propuesta se basa en la implementación y entrenamiento de una 3D-CNN a través de un generador que garantiza la correcta aleatoriedad de los datos de entrenamiento.

El uso de información de profundidad permite determinar las acciones realizadas por diferentes personas en la escena, preservando su privacidad, al no ser posible reconocer la identidad de cada individuo, por lo que puede ser utilizado incluso en entornos en los que existan requisitos relacionadas con la privacidad, por ejemplo hospitales, centros de mayores, etc.

La propuesta presentada ha sido evaluada experimentalmente de forma exhaustiva, empleando bases de datos públicas (disponible para investigación) descritas en el apartado 5.1.1. Los resultados obtenidos han permitido validar la propuesta al tener una precisión global en la clasificación de acciones, empleando la base de datos previamente entrenada, superior al 94 %, a pesar de la complejidad de la base de datos. Además, se ha demostrado la robustez y la correcta generalización de la red neuronal a través de pruebas experimentales con una base de datos con un escenario y puntos de vista distintos a la empleada para entrenamiento alcanzando una precisión superior al 67 %.

El desarrollo de este TFG ha permitido la publicación de un artículo [62] en el Congreso Nacional SAAEI 2019.

## 6.2 Líneas futuras

En esta sección se presentan las posibles futuras líneas de investigación del sistema de detección de acciones desarrollado. Las más importantes se exponen a continuación:

- **Optimización de las técnicas de entrenamiento:** con el objetivo fundamental de incrementar la precisión categórica en la tarea de la detección de acciones. Como principal técnica para mejorar el entrenamiento, se debe considerar el *data augmentation* a través de la creación de una base de datos sintética aplicando la modificación de los píxeles de las secuencias de video que permitan simular

la acción a diferentes distancias de la cámara, sin modificar su modelo mediante el redimensionado pertinente.

- **Redes Neuronales en paralelo:** la implementación de 3D-CNN en paralelo permitiría el reconocimiento de acciones en situaciones más realistas de solapamiento de acciones. Cada una de ellas, realizaría las operaciones de convolución sobre secuencias de vídeo de distinto tamaño y mediante una capa de decisión, se asignaría un peso a la salida de cada una de ellas. La implementación de las redes en paralelo sería inmediata, pues la arquitectura sería idéntica, excepto la necesaria modificación de la capa de entrada de cada una de ellas.

- **Sistema en tiempo real:** se podría abordar el desarrollo de un sistema que permita la detección de acciones en tiempo real.

- **Grabación y evaluación de una base de datos propia:** la grabación y creación de una base de datos que incluya únicamente acciones que pertenezcan a un contexto de la seguridad y la video-vigilancia.

# Capítulo 7

# Pliego de condiciones

## 7.1 Introducción

En este apartado se evalúan las condiciones necesarias, tanto *hardware* como *software*, para un correcto funcionamiento del sistema desarrollado en este TFG.

## 7.2 Requisitos de hardware

- GPU GeForce NVIDIA 1080GTX con 8GB de memoria.
- 16 GB de RAM DDR2 a 800 MHz o superior.
- Procesador de 64 bits multi-core.
- Al menos 80GB de memoria libre en disco duro

El sistema de detección de acciones que se propone hace uso de hilos independientes en la preparación de los datos de entrenamiento por parte del generador. Es por esta razón que se recomienda sistemas multiprocesador, que permitan realizar las tareas de cada hilo de manera independiente.

El entrenamiento de una red neuronal es un proceso largo y costoso que precisa de una gran cantidad de datos. Por esta razón es necesario la utilización de un volumen importante de memoria RAM, se recomienda que la cifra de partida sean 8 Gb o superior.

Se recomienda tener al menos 80 GB de disco duro libre para poder almacenar las bases de datos empleadas y la instalación de las librerías requeridas, así como programas como Matlab.

## 7.3 Requisitos de software

- Utilización de un sistema operativo Ubuntu 16.04.
- Librería `Keras 2.1.2`.
- Librería `TensorFlow 1.4.0`.
- Python 3.5.

- Pycharm Community
- Procesador de Latex
- Matlab

En este apartado *software* se recomienda utilizar el sistema operativo Ubuntu 16.04 al ser LTS, y proporcionar la compatibilidad con las librerias de Keras que permiten la definición de las capas de una red neuronal de forma sencilla.

El empleo de MatLab está destinado a la preparación de las secuencias de vídeo de acciones y a la posterior extracción de resultados.

# Capítulo 8

# Presupuesto

## 8.1 Costes de equipamiento

### 8.1.1 Equipamiento hardware utilizado:

| Concepto | Cantidad | Coste unitario | Subtotal(euros) |
|---|---|---|---|
| Ordenador i7 + NVIDIA GTX1080 | 1 | 2000 | 2000 |
| Camara Tof Kinect v2 | 1 | 176,71 | 176,71 |
| Coste Total | | | 2176,71 |

### 8.1.2 Recursos software utilizados:

| Concepto | Cantidad | Coste unitario | Subtotal(euros) |
|---|---|---|---|
| Ubuntu 16.04 LTS | 1 | 0 | 0 |
| Librerias Opencv Python 3.1.0 | 1 | 0 | 0 |
| Librerias TensorFlow 1.4.0 | 1 | 0 | 0 |
| Librerias Keras 2.10 | 1 | 0 | 0 |
| Matlab | 1 | 2000 | 2000 |
| Texstudio | 1 | 0 | 0 |
| Total | | | 2000 |

## 8.2 Costes Mano de obra

| Concepto | Cantidad | Coste unitario | Subtotal(euros) |
|---|---|---|---|
| Programacion y Desarrollo de Software | 250 | 60euros/hora | 15000 |
| Mecanografiado de documentación, manuales y tutoriales | 60 | 15 euros/hora | 900 |
| Total | | | 15900 |

## 8.3 Costes Totales

| Concepto | Subtotal |
|---|---|
| Hardware | 2176,71 |
| Software | 2000 |
| Mano de obra | 15900 |
| Total | 20076,71 |

El importe total del presupuesto asciende a la cantidad de: VEINTE MIL SETENTA Y SEIS EUROS

En Alcalá de Henares, a 20 de junio de 2019.

Sergio de López Diz Graduado en Ingeniería Electrónica de Comunicaciones

# Bibliografía


[1] S. Beak, Z. Shi, M. Kawade, and T.-K. Kim, "Kinematic-aware random forest for static and dynamic action recognition from depth sequences," 07 2016.

[2] T. Bagautdinov, A. Alahi, F. Fleuret, P. Fua, and S. Savarese, "Social scene understanding: End-to-end multi-person action localization and collective activity recognition," *30Th Ieee Conference On Computer Vision And Pattern Recognition (Cvpr 2017)*, pp. 10. 3425–3434, 2017. [Online]. Available: http://infoscience.epfl.ch/record/230241

[3] K. Soomro and M. Shah, "Unsupervised action discovery and localization in videos," in *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct 2017, pp. 696–705.

[4] S. Zaidenberg, B. Boulay, and F. Bremond, "A generic framework for video understanding applied to group behavior recognition," in *9th IEEE International Conference on Advanced Video and Signal-Based Surveillance (AVSS 2012)*, ser. Advanced Video and Signal Based Surveillance, IEEE Conference on, IEEE Computer Society. Beijing, China: IEEE Computer Society, Sep. 2012, pp. 136 –142. [Online]. Available: https://hal.inria.fr/hal-00702179

[5] "Real Sense Stereo Camera," https://realsense.intel.com/stereo-depth-vision-basics/.

[6] S. Hussmann, T. Ringbeck, and B. Hagebeuker, "A performance review of 3d tof vision systems in comparison to stereo vision systems," in *Stereo vision*. InTechOpen, 2008.

[7] B. Altakrouri, "Ambient assisted living with dynamic interaction ensembles," Ph.D. dissertation, 08 2014.

[8] J. Jiao, L. Yuan, W. Tang, Z. Deng, and Q. Wu, "A post-rectification approach of depth images of kinect v2 for 3d reconstruction of indoor scenes," *ISPRS International Journal of Geo-Information*, vol. 6, p. 349, 11 2017.

[9] A. Islam, M. A. Hossain, and Y. M. Jang, "Interference mitigation technique for time-of-flight (tof) camera," *2016 Eighth International Conference on Ubiquitous and Future Networks (ICUFN)*, pp. 134–139, 2016.

[10] M. Hansard, S. Lee, O. Choi, and R. Horaud, *Time of Flight Cameras: Principles, Methods, and Applications*, 10 2012.

[11] "Modelo neuronal de McCulloch-Pitts," disponible online: http://www.cs.us.es/~fsancho/?e=72 (Último acceso 27/05/2019).

[12] "Guide to Neural Networks and Deep Learning," disponible online: https://skymind.ai/wiki/neural-network (Último acceso 24/04/2019).

[13] A. Shahroudy, J. Liu, T.-T. Ng, and G. Wang, "Ntu rgb+d: A large scale dataset for 3d human activity analysis," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[14] "NTU RGB+D Action Recognition dataset," disponible online: http://rose1.ntu.edu.sg/datasets/actionrecognition.asp (Último acceso 24/01/2019).

[15] R. Poppe, "A survey on vision-based human action recognition," *Image and vision computing*, vol. 28, no. 6, pp. 976–990, 2010.

[16] D. Weinland, R. Ronfard, and E. Boyer, "A survey of vision-based methods for action representation, segmentation and recognition," *Computer vision and image understanding*, vol. 115, no. 2, pp. 224–241, 2011.

[17] C. Chen, R. Jafari, and N. Kehtarnavaz, "A survey of depth and inertial sensor fusion for human action recognition," *Multimedia Tools and Applications*, vol. 76, no. 3, pp. 4405–4425, 2017.

[18] S. Sadanand and J. J. Corso, "Action bank: A high-level representation of activity in video," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 1234–1241.

[19] J. Zhang, Y. Han, J. Tang, Q. Hu, and J. Jiang, "Semi-supervised image-to-video adaptation for video action recognition," *IEEE transactions on cybernetics*, vol. 47, no. 4, pp. 960–973, 2017.

[20] J. Han, L. Shao, D. Xu, and J. Shotton, "Enhanced computer vision with microsoft kinect sensor: A review," *IEEE transactions on cybernetics*, vol. 43, no. 5, pp. 1318–1334, 2013.

[21] J. Sell and P. O'Connor, "The Xbox one system on a chip and Kinect sensor," *Micro, IEEE*, vol. 34, no. 2, pp. 44–53, Mar 2014.

[22] N. Ashraf, C. Sun, and H. Foroosh, "View invariant action recognition using projective depth," *Computer Vision and Image Understanding*, vol. 123, pp. 41–52, 2014.

[23] A.-A. Liu, W.-Z. Nie, Y.-T. Su, L. Ma, T. Hao, and Z.-X. Yang, "Coupled hidden conditional random fields for rgb-d human action recognition," *Signal Processing*, vol. 112, pp. 74 – 82, 2015, signal Processing and Learning Methods for 3D Semantic Analysis. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0165168414004022

[24] P. Khaire, P. Kumar, and J. Imran, "Combining cnn streams of rgb-d and skeletal data for human activity recognition," *Pattern Recognition Letters*, vol. 115, pp. 107 – 116, 2018, multimodal Fusion for Pattern Recognition. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167865518301636

[25] R. Lange and P. Seitz, "Solid-state time-of-flight range camera," *Quantum Electronics, IEEE Journal of*, vol. 37, no. 3, pp. 390–397, Mar 2001.

[26] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105. [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[27] Y. Xu, J. Du, L.-R. Dai, and C.-H. Lee, "A regression approach to speech enhancement based on deep neural networks," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 1, pp. 7–19, 2015.

[28] A. Ullah, J. Ahmad, K. Muhammad, M. Sajjad, and S. W. Baik, "Action recognition in video sequences using deep bi-directional lstm with cnn features," *IEEE Access*, vol. 6, pp. 1155–1166, 2018.

[29] S. Das, M. Koperski, F. Bremond, and G. Francesca, "A fusion of appearance based cnns and temporal evolution of skeleton with LSTM for daily living action recognition," *CoRR*, vol. abs/1802.00421, 2018. [Online]. Available: http://arxiv.org/abs/1802.00421

[30] S. Ji, W. Xu, M. Yang, and K. Yu, "3d convolutional neural networks for human action recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 1, pp. 221–231, Jan 2013.

[31] H. Yang, C. Yuan, B. Li, Y. Du, J. Xing, W. Hu, and S. J. Maybank, "Asymmetric 3d convolutional neural networks for action recognition," *Pattern Recognition*, vol. 85, pp. 1 – 12, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0031320318302632

[32] L. Xia, C. Chen, and J. Aggarwal, "View invariant human action recognition using histograms of 3d joints," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on*. IEEE, 2012, pp. 20–27.

[33] B. Fosty, C. F. Crispim-Junior, J. Badie, F. Bremond, and M. Thonnat, "Event Recognition System for Older People Monitoring Using an RGB-D Camera," in *ASROB - Workshop on Assistance and Service Robotics in a Human Environment*, Tokyo, Japan, Nov. 2013. [Online]. Available: https://hal.inria.fr/hal-00904002

[34] L. Xia, C. Chen, and J. K. Aggarwal, "View invariant human action recognition using histograms of 3d joints," in *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, June 2012, pp. 20–27.

[35] H. Foroughi, A. Naseri, A. Saberi, and H. Sadoghi Yazdi, "An eigenspace-based approach for human fall detection using integrated time motion image and neural network," in *2008 9th International Conference on Signal Processing*, Oct 2008, pp. 1499–1503.

[36] E. Bermejo Nievas, O. Deniz Suarez, G. Bueno García, and R. Sukthankar, "Violence detection in video using computer vision techniques," in *Computer Analysis of Images and Patterns*, P. Real, D. Diaz-Pernil, H. Molina-Abril, A. Berciano, and W. Kropatsch, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 332–339.

[37] S. Ji, W. Xu, M. Yang, and K. Yu, "3d convolutional neural networks for human action recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 1, pp. 221–231, Jan 2013.

[38] G. Zhu, L. Zhang, L. Mei, Jie Shao, Juan Song, and Peiyi Shen, "Large-scale isolated gesture recognition using pyramidal 3d convolutional networks," in *2016 23rd International Conference on Pattern Recognition (ICPR)*, Dec 2016, pp. 19–24.

[39] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell, "Long-term recurrent convolutional networks for visual recognition and description," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[40] B. Mahasseni and S. Todorovic, "Regularizing long short term memory with 3d human-skeleton sequences for action recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016, pp. 3054–3062.

[41] P. Wang, W. Li, Z. Gao, C. Tang, and P. O. Ogunbona, "Depth pooling based large-scale 3-d action recognition with convolutional neural networks," *IEEE Transactions on Multimedia*, vol. 20, no. 5, pp. 1051–1061, May 2018.

[42] Y. Hou, S. Wang, P. Wang, Z. Gao, and W. Li, "Spatially and temporally structured global to local aggregation of dynamic depth information for action recognition," *IEEE Access*, vol. 6, pp. 2206–2219, 2018.

[43] P. Wang, W. Li, P. Ogunbona, J. Wan, and S. Escalera, "Rgb-d-based human motion recognition with deep learning: A survey," *Computer Vision and Image Understanding*, vol. 171, pp. 118 – 139, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1077314218300663

[44] M. L. Álvarez Pastor, "Configuración y ejecución de algoritmos de visión artificial en la tarjeta nvidia jetson tk1 devkit," Master's thesis, Trabajo Fin de Grado. Universidad de Alcalá, 2017.

[45] J. Smisek, M. Jancosek, and T. Pajdla, *3D with Kinect*. London: Springer London, 2013, pp. 3–25. [Online]. Available: https://doi.org/10.1007/978-1-4471-4640-7_1

[46] "Asus Xtion-Pro," disponible online: https://www.asus.com/es/3D-Sensor/Xtion_PRO/ (Último acceso 24/04/2019).

[47] "Cámara con sensor de profundidad intel realsense serie d400," disponible online: https://software.intel.com/es-es/realsense/d400 (Último acceso: 03/06/2019).

[48] D. Jiménez, D. Pizarro, M. Mazo, and S. Palazuelos, "Modeling and correction of multipath interference in time of flight cameras," *Image and Vision Computing*, vol. 32, no. 1, pp. 1 – 13, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0262885613001650

[49] D. Jimenez, D. Pizarro, and M. Mazo, "Single frame correction of motion artifacts in pmd-based time of flight cameras," *Image and Vision Computing*, vol. 32, no. 12, pp. 1127–1143, 2014.

[50] M. Baptista Ríos, "Anomalous behaviour detection in video surveillance scenes," Master's thesis, Escuela Politécnica Superior. Universidad de Alcalá, 2017.

[51] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, G. Gordon, D. Dunson, and M. Dudík, Eds., vol. 15. Fort Lauderdale, FL, USA: PMLR, 11–13 Apr 2011, pp. 315–323. [Online]. Available: http://proceedings.mlr.press/v15/glorot11a.html

[52] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 06 2014.

[53] L. Bottou, "Large-scale machine learning with stochastic gradient descent," *Proc. of COMPSTAT*, 01 2010.

[54] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014. [Online]. Available: http://arxiv.org/abs/1412.6980

[55] T. Dozat, "Incorporating nesterov momentum into," 2015.

[56] J. C. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, pp. 2121–2159, 07 2011.

[57] S. Ruder, "An overview of gradient descent optimization algorithms," *CoRR*, vol. abs/1609.04747, 2016. [Online]. Available: http://arxiv.org/abs/1609.04747

[58] D. Masters and C. Luschi, "Revisiting small batch training for deep neural networks," *CoRR*, vol. abs/1804.07612, 2018. [Online]. Available: http://arxiv.org/abs/1804.07612

[59] K. Soomro, A. R. Zamir, and M. Shah, "UCF101: A dataset of 101 human actions classes from videos in the wild," *CoRR*, vol. abs/1212.0402, 2012. [Online]. Available: http://arxiv.org/abs/1212.0402

[60] L. Bottou, "Large-scale machine learning with stochastic gradient descent," *Proc. of COMPSTAT*, 01 2010.

[61] D. M. Harris and S. L. Harris, "3 - sequential logic design," in *Digital Design and Computer Architecture (Second Edition)*, second edition ed., D. M. Harris and S. L. Harris, Eds. Boston: Morgan Kaufmann, 2013, p. 129. [Online]. Available: http://www.sciencedirect.com/science/article/pii/B9780123944245000033

[62] S. de López Diz, C. Losada Gutiérrez, and D. Fuentes Jiménez, "Detección de acciones a partir de información de profundidad mediante redes neuronales convolucionales," *SAAEI Córdoba*, 07 2019.

# Apéndice A

# Manual de usuario

## A.1 Introducción

Este manual de usuario especifica cada uno de los pasos que se deben seguir para el correcto empleo del sistema de detección de acciones implementado en este TFG.

## A.2 Requisitos previos

El sistema de reconocimiento de acciones se ha desarrollado en el lenguaje interpretado *Python* en un entorno *Linux* empleando la versión *Ubuntu 16.04 LTS*. Para el correcto funcionamiento del sistema se recomienda el uso de una versión *Ubuntu 16.04* o superior junto con la instalación de los programas y librerías que se exponen a continuación:

- **Python 3.5:** se debe instalar la versión 3.5 de *Python* que permita la programación del sistema desarrollado.
- **Librería TensorFlow 1.4.0:** *TensorFlow* es una plataforma de código abierto *end-to-end* para *machine learning*. Presenta un ecosistema de herramientas y librerías que permite la construcción de aplicaciones potentes centradas en el *machine learning*.
- **Librería Keras 2.1.2:** *Keras* es una librería de *TensorFlow* que presenta una API de alto nivel que permite la construcción y el entrenamiento de modelos de *Deep Learning*. Esta librería permite la construcción y posterior entrenamiento de la 3D-CNN que forma el sistema de detección de acciones.
- **Librerías Opencv 3.0.0:** *Opencv* son un conjunto de librerías que incluyen gran diversidad de funciones relacionadas con el contexto de la visión artificial. En este TFG permite el procesado de las secuencias de imágenes y de los datos necesarios para el entrenamiento de la red neuronal.
- **Pycharm Community 2018:** se recomienda la instalación de la interfaz *Pycharm Community* que permite trabajar con mayor comodidad en un entorno *Python*.
- **MatLab:** permite de una forma simple y cómoda el análisis gráfico de los resultados experimentales obtenidos mediante la creación de una matriz de confusión.

## A.3 Estructura del programa

El sistema desarrollado consta de varios archivos que desarrollan una determinada función en el sistema de detección de acciones desarrollado en este TFG. Todos ellos, deben encontrarse dentro de una misma carpeta para la correcta compilación y ejecución del programa. Además de los propios archivos de código fuente, se deben incluir 3 carpetas que contengan las secuencias de imágenes de profundidad para el entrenamiento, validación y test de la red neuronal. Dentro de cada una de ellas, se deben organizar nuevamente en carpetas todos los vídeos que realicen una misma acción que permita el etiquetado y preparación de los datos a introducir a la red. A continuación se exponen los archivos fuentes más importantes y su función correspondiente:

- **3dcnn.py:** Alberga la mayor funcionalidad del sistema desarrollado. Incluye las funciones generadores necesarias para el entrenamiento, así como la generación de gráficas de entrenamiento, almacenamiento de la arquitectura y pesos de la red, creación de ficheros de resultados que permiten la evaluación del sistema...etc. En el mismo, se permite seleccionar si se desea realizar un entrenamiento de la red, cargando o no pesos pre-entrenados, o ejecutar únicamente la etapa de test con los datos correspondientes. Además, en la etapa de entrenamiento se permite la personalización en la selección del tamaño de *batch* o el número de épocas.
- **videoto3D.py:** Alberga la función que permite la correcta adaptación de los datos a la capa de entrada implementada en la red neuronal.

## A.4 Ejecución del programa

En función de las etapas seleccionadas para su ejecución, el programa se comporta de forma diferente.

Si se selecciona realizar únicamente la etapa de entrenamiento, como resultado de dicho proceso, se guardará la arquitectura de la red neuronal en formato *json* y los mejores pesos obtenidos durante las épocas elegidas. Además, se generarán una gráficas que muestran la función de pérdidas y la precisión obtenida en cada época, permitiendo un análisis a posteriori del comportamiento de la red.

Si se selecciona realizar únicamente la etapa de test, se generará un archivo de test que permita la evaluación del comportamiento de la red, así como la visualización de la secuencia de vídeo de profundidad obtenida con las tres acciones más probables proporcionadas por la red.

Si se selecciona realizar ambas etapas, test y entrenamiento, se generarán el conjunto de archivos comentados en las etapas ejecutadas de forma independiente.

### A.4.1 Resultados de la aplicación

Una vez se haya iniciado la aplicación seleccionando la etapa de test, se prepararán todos los datos destinados a ese fin (almacenados en la carpeta de test) y se introducirán en la red generando un fichero de resultados donde se indique el código de acción etiquetado, el predicho y la máxima probabilidad obtenida. Un ejemplo del fichero de resultados generado se muestra en la figura A.1.

Tras la generación del fichero, se reproduce uno a uno los vídeos de test con el objetivo de observar de manera visual los vídeos de acción de profundidad introducidos a la red. Una vez reproducidos, se muestran ordenadas las 3 acciones con mayor probabilidad generadas como salida de la red implementada. La aplicación se queda bloqueada hasta la pulsación de una tecla, momento en el que se empezará a

Open

| Clase Real | Clase Predicción | Precisión |
|---|---|---|
| 0 | 0 | 0.9989970326423645 |
| 0 | 0 | 0.9999864101409912 |
| 0 | 0 | 1.0 |
| 0 | 0 | 0.9994799494743347 |
| 0 | 0 | 0.9999780654907227 |
| 0 | 1 | 0.5807621479034424 |
| 0 | 0 | 0.9817819595336914 |
| 0 | 0 | 0.9736959934234619 |
| 0 | 0 | 0.9999126195907593 |
| 0 | 0 | 0.9727803468704224 |
| 0 | 0 | 0.9958139061927795 |
| 0 | 0 | 0.5276029706001282 |
| 0 | 0 | 0.9999406337738037 |
| 0 | 0 | 0.9951890707015991 |
| 0 | 0 | 0.6973166465759277 |
| 0 | 0 | 0.999916672706604 |
| 0 | 0 | 0.4604456126689911 |
| 0 | 0 | 0.9469471573829651 |
| 0 | 0 | 0.9697784781455994 |
| 0 | 5 | 0.8575195670127869 |
| 0 | 0 | 0.9980286955833435 |
| 0 | 0 | 0.9999765157699585 |
| 0 | 0 | 0.9961680769920349 |
| 0 | 0 | 0.9123003482818604 |
| 0 | 0 | 0.9999997615814209 |
| 0 | 0 | 1.0 |
| 0 | 0 | 0.9777141213417053 |
| 0 | 0 | 1.0 |
| 0 | 5 | 0.7835609912872314 |
| 0 | 0 | 0.9927270412445068 |
| 0 | 7 | 0.6609809994697571 |
| 0 | 0 | 0.9999904632568359 |

Figura A.1: Ejemplo de fichero de resultados.

reproducir la siguiente acción. En la figura A.2 se refleja un ejemplo de reproducción de una acción junto con sus correspondientes probabilidades.


Figura A.2: Ejemplo de reproducción de acción junto con las 3 acciones más probables.

Universidad de Alcalá
Escuela Politécnica Superior



ESCUELA POLITECNICA
SUPERIOR

Universidad
de Alcalá