Implementación de un sistema de seguimiento de personas y reconocimiento de acciones humanas en vídeo mediante aprendizaje profundo.
Implementation of a people tracking system and recognition of human actions on video using deep learning

Trabajo de Fin de Máster
Curso 2019–2020

Autor
Javier Antón Alonso

Director
Gonzalo Pajares Martinsanz

Convocatoria: Junio 2020

Calificación: 10 - Sobresaliente

Máster en Ingeniería Informática
Facultad de Informática
Universidad Complutense de Madrid

Implementación de un sistema de seguimiento de personas y reconocimiento de acciones humanas en vídeo mediante aprendizaje profundo.
Implementation of a people tracking system and recognition of human actions on video using deep learning

Trabajo de Fin de Máster en Ingeniería Informática
Departamento de Ingeniería del Software e Inteligencia Artificial

Autor
Javier Antón Alonso

Director
Gonzalo Pajares Martinsanz

Convocatoria: Junio 2020
Calificación: 10 - Sobresaliente

Máster en Ingeniería Informática
Facultad de Informática
Universidad Complutense de Madrid

24 de Julio de 2020

Autorización de difusión

El abajo firmante, matriculado en el **Máster en Ingeniería en Informática** de la Facultad de Informática, autoriza a la Universidad Complutense de Madrid (UCM) a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a su autor el presente Trabajo Fin de Máster: **"Implementación de un sistema de seguimiento de personas y reconocimiento de acciones humanas en vídeo mediante aprendizaje profundo (Implementation of a people tracking system and recognition of human actions on video using deep learning)"**, realizado durante el curso académico **2019-2020** bajo la dirección de **GONZALO PAJARES MARTINSANZ** en el Departamento de **INGENIERÍA DEL SOFTWARE E INTELIGENCIA ARTIFICIAL**, y a la Biblioteca de la UCM a depositarlo en el Archivo Institucional E-Prints Complutense con el objeto de incrementar la difusión, uso e impacto del trabajo en Internet y garantizar su preservación y acceso a largo plazo.

Javier Antón Alonso

24 de julio de 2020

Dedicatoria

A mis padres, mi hermano, mi novia Ana, y a mis amigos, pero especialmente a mis abuelos, que ya no están con nosotros.

Agradecimientos

Quiero agradecer los ánimos y la ayuda recibida a una gran cantidad de personas, gracias a las cuales este trabajo se ha podido llevar a cabo.

Primeramente , quiero agradecer a mi tutor Gonzalo Pajares por darme la oportunidad de realizar este TFM, por su ayuda y su apoyo durante la realización del mismo.

Además quiero dar las gracias a mi hermano, mi novia Ana, mis amigos, mis compañeros de Foqum, y en especial a mis padres, que me han dado fuerzas y prestado su ayuda siempre que la necesité.

Agradecer también a todos los profesores que he tenido a lo largo de la carrera y el máster, que con su paciencia y su ayuda, me han ayudado a adquirir los conocimientos necesarios en mi formación.

Resumen

El reconocimiento de la actividad en vídeo ha sido objeto de diversos esfuerzos de investigación, debido a la importancia que podría conllevar su uso en el mundo real. Videovigilancia o robótica son dos ramas muy beneficiadas con los avances en este campo, especialmente en el campo de la robótica las tareas relacionadas con la navegación autónoma o la interacción de los robots con seres humanos. Aunque éstos no serían las únicas aplicaciones, ya que la extracción de información inherente a ciertas situaciones nos llevaría a poder hacer análisis detalladas del comportamiento social humano, abriendo un amplio camino por delante.

El objetivo de este proyecto es el de implementar un sistema capaz de detectar, realizar seguimiento y reconocer las acciones realizadas por humanos en vídeos. Destacando especialmente 3 partes diferenciadas en el proyecto:

- Detección de personas, desde el punto de vista de la detección de objetos mediante técnicas de aprendizaje profundo, estudiando y usando algoritmos como Faster-RCNN y YOLO.
- Diseño e implementación de un sistema de rastreo, mediante el uso de algoritmos, como el Filtro de Kalman y el algoritmo Húngaro.
- Reconocimiento de acciones humanas, para lo cual se usan las técnicas de aprendizaje profundo con mejores resultados en el reconocimiento de actividades en vídeo, las redes convolucionales i3D y las redes resnet 3D, así como la puesta en práctica de algoritmos de procesamientos de flujos ópticos en vídeo.

Palabras clave

Reconocimiento de actividades humanas, aprendizaje profundo, red neuronal, arquitectura de dos flujos, redes convolucionales, redes i3D, resnet, flujo óptico, procesamiento de vídeo.

Abstract

The recognition of the video activity has been the subject of various research efforts, due to the importance that its use in the real world could entail. Video surveillance or robotics are two branches that benefit greatly from advances in this field, especially in the field of robotics, tasks related to autonomous navigation or the interaction of robots with human beings. Although these would not be the only applications, since the extraction of information inherent in certain situations would lead us to be able to make detailed analyzes of human social behavior, opening a wide path ahead.

The objective of this project is to implement a system capable of detecting, tracking and recognizing the actions carried out by humans in videos. Especially highlighting 3 different parts in the project:

- Detection of humans, testing and using algorithms like Faster-RCNN and YOLO.
- Design and implementation of a tracking system, through the use of algorithms, such as the Kalman Filter or the Hungarian algorithm, among others.
- Recognition of human actions, for which I will use some of the deep learning techniques with better results in the recognition of video activities, i3D convolutional networks and 3D resnet networks, as well as implement different algorithms for processing optical flows in video. Achieving this will also allow me to carry out a comparison of which achieve the best results.

Keywords

Activity recognition, deep learning, neural network, two stream architecture, convolutional networks, i3D networks, resnet, python, opencv, optical flow, video processing.

Índice

Índice de figuras

Capítulo 1

Introducción

"Algunas personas denominan a esta tecnología inteligencia artificial, cuando en realidad lo que va a permitir es que aumente la nuestra propia"
— Gin Rometty

1.1. Motivación

Nosotros, como seres humanos, siempre hemos tratado de obtener la mayor información posible del entorno que nos rodea, sea de forma directa o indirecta. De esta manera el ser humano ha buscado siempre fuentes de información, de las que aprender, de las que sacar ventaja. Pero no ha sido hasta los avances en computación de las ultimas décadas, que estas tareas de obtención y análisis de información han podido ser, en cierta medida, cedidas a las máquinas.

Siendo justos, la velocidad del razonamiento humano secuencial es baja en comparación con los ordenadores, por lo cual se puede pensar que cualquier problema plasmado en forma de algoritmo, y dado a un ordenador podría ser resuelto siempre, o al menos un elevado número de veces. No obstante hay algo en lo que el ser humano supera a la máquina, la posibilidad de formular juicios y obtener información dentro de contextos de los que no sabe nada previamente.

Por el contrario, las máquinas requieren de gran cantidad de información para entrenar, especialmente si aplicamos aprendizaje profundo. Pero una vez la máquina ha aprendido, podrá detectar esta información, en este caso las actividades realizadas en un vídeo, de forma más precisa que un humano.

La posibilidad de detectar qué acción o actividad se está desarrollando en un vídeo, se auto-alimenta de la motivación surgida a raíz de pensar en los usos que se le podrían dar. Por ejemplo el poder etiquetar automáticamente vídeos resultaría de gran utilidad para poder agilizar las búsquedas en bases de datos de vídeos, al detectar por ejemplo un bateo de baseball, alguien tocando la guitarra o alguien afeitándose. Su aplicación en videovigilancia automática es una vertiente con muchas posibilidades, tanto para detectar actividades criminales como incluso poder llegar a predecirlas.

Aquí es donde entra el aprendizaje profundo, que en conjunto con técnicas de visión

computarizada, nos permiten realizar la detección de estas actividades.

1.2. Objetivos

El objetivo final de este proyecto consiste en:

Diseñar e implementar un sistema de vídeo vigilancia capaz de detectar humanos, rastrearlos como entidades independientes prediciendo sus movimientos y reconocer las acciones que están realizando.

Los objetivos intermedios necesarios para poder cumplirlo, se dividen en varios grupos:.

- Investigar las técnicas usadas en reconocimiento de vídeos, utilizando en particular la arquitectura de dos flujos de redes convolucionales i3D y la arquitectura Resnet 3D.
- Investigar e implementar varias técnicas de obtención de flujos ópticos de vídeos.
- Investigar y utilizar las técnicas mas avanzadas en detección de objetos en imágenes, buscando primar la velocidad de cómputo en tiempo real.
- Implementar un sistema de rastreo de humanos dentro de un plano de vídeo, capaz de identificar sus movimientos pasados y predecir los futuros.

Las tareas abordadas durante el desarrollo del trabajo se clasifican dentro de los siguientes campos computacionales:

- **Inteligencia artificial**, concretamente la rama del aprendizaje automático, en concreto del aprendizaje profundo.
- **Visión por computador**, en lo relevante a las técnicas de obtención de flujos ópticos de vídeos.

1.3. Plan de trabajo

La realización de este proyecto fue dividida en un comienzo en las siguientes fases, no obstante, no siempre se ha podido seguir este plan de forma estricta, y bajo ciertas circunstancias han tenido que ser reordenadas y revisadas, sin que ello haya afectado al desarrollo esencial de las mismas. Las fases son:

- Fase 0: Definición a grandes rasgos del sistema a realizar y de las partes necesarias que lo conformarían.
- Fase 1: Investigación de técnicas y algoritmos para detección de objetos en vídeos.
- Fase 2: Diseño e implementación de un sistema capaz de detectar humanos en vídeos, según las necesidades del proyecto.
- Fase 3: Investigación de técnicas y algoritmos para rastrear el mayor número posible de entidades independientes dentro de un vídeo, según las necesidades del proyecto.

- Fase 4: Diseño e implementación de un sistema de rastreo de humanos en vídeos, capaz de asignar identificadores únicos a cada uno de los humanos, de almacenar sus movimientos pasados y de predecir sus movimientos futuros.
- Fase 5: Investigación de técnicas y algoritmos de reconocimiento de acciones humanas en vídeos.
- Fase 6: Diseño e implementación de un sistema capaz de reconocer las acciones humanas presentes en vídeos, primando la velocidad.
- Fase 7: Integración de los componentes desarrollados anteriormente con el objetivo de formar el sistema de vídeovigilancia inteligente.
- Fase 8: Realización de pruebas sobre el sistema final.
- Fase 9: Escritura de la memoria del proyecto.

Las fases mencionadas constituyen intrínsecamente la principal aportación personal al trabajo desarrollado, destacando el estudio de las técnicas relativas a detección de personas en vídeos y la aplicación de técnicas de aprendizaje profundo, con el fin de identificar las más apropiadas, para realizar el diseño e integración de las mismas en un sistema conjunto, que permite el análisis de resultados.

1.4. Organización de la memoria

La memoria está organizada en los capítulos que se indican a continuación.

- **Capítulo 1: Introducción**

  Es el capítulo introductorio al proyecto, en él se describe la motivación que subyace del desarrollo de este trabajo, los objetivos a conseguir durante la resolución del mismo, así como el plan de trabajo a seguir y la organización de la memoria.

- **Capítulo 2: Trabajo previo**

  A lo largo de este capítulo se describen los aspectos mas relevantes de la investigación del estado del arte de las técnicas existentes en el ámbito del reconocimiento de actividades humanas en vídeos, desde su origen hasta la actualidad. Finalmente se exponen las tecnologías investigadas para el desarrollo del proyecto.

- **Capítulo 3: Flujos ópticos**

  Entrando ya en el campo de la visión artificial, en este capítulo se analiza la teoría del movimiento en su variante psicológica, de forma introductoria para poder entender a continuación los conceptos en los que se sustenta la detección del movimiento, según el método del flujo óptico de un vídeo.

- **Capítulo 4: Redes neuronales artificiales**

  Este capítulo entra en el mundo de las redes neuronales, comenzando desde una aproximación biológica, para poco a poco ir comprendiendo los conceptos inherentes a las mismas. El foco principal está puesto en torno a las redes neuronales convolucionales, y en las capas que las conforman, no obstante, también se describen otros tipos de redes utilizadas en este ámbito.

- **Capítulo 5: Metodología de trabajo**

  En este capítulo se exponen las metodologías seguidas durante la realización del proyecto, así como algunas de las herramientas que han resultado útiles de cara a organizar el trabajo.

- **Capítulo 6: Sistema de videovigilancia**

  El propósito principal de este capítulo es detallar la construcción del sistema desarrollado, explicando su funcionamiento general y los componentes que lo conforman.

- **Capítulo 7: Resultados**

  En este capítulo se encuentran recogidos, incluyendo tablas e imágenes, los resultados obtenidos por la realización de diferentes pruebas.

- **Capítulo 8: Conclusiones y trabajo futuro**

  Este capítulo contiene las conclusiones obtenidas tras reflexionar sobre los resultados de las pruebas realizadas, así como tener en cuenta posibles trabajos futuros que pudieran derivarse.

Capítulo 2

Trabajos previos

2.1. Introducción

Las posibles utilidades del reconocimiento de actividades comenzaron a generar interés en los científicos durante los años 80, ya que tiene aplicaciones tanto en actividades de vigilancia como en medicina o sociología.La posibilidad de crear un sistema capaz de detectar de forma automática, por ejemplo, que se está cometiendo un crimen, permitirá llevar a cabo medidas de contención rápidamente.

Para poder llevar a cabo estas detecciones se han logrado grandes progresos gracias a tecnologías, tales como la extracción de los flujos ópticos de un vídeo y diferentes aproximaciones de aprendizaje profundo. No obstante, pese al gran éxito de las arquitecturas de aprendizaje profundo en clasificación de imágenes, los avances en clasificación de vídeos se han producido mas lentamente, sin haberse conseguido aún precisiones tan altas como en imágenes, siendo ahora mismo un campo bastante abierto a la investigación.

Las razones principales del avance tan lento en este campo son las siguientes:

- Coste computacional: se requiere de enormes cantidades de tiempo para poder entrenar diferentes arquitecturas de aprendizaje profundo, ya que el número de parámetros que se maneja es inmenso.
- Los datos con los que se cuentan para entrenar las redes son muy pocos, y no existe un dataset estándar para vídeo como sería el caso de Imagenet en clasificación de imágenes, ya que almacenar y etiquetar esas grandes cantidades de vídeos supone una gran dificultad.
- A diferencia de la clasificación de imágenes, no solo se necesita la información espacial, sino también capturar el contexto en el que se desarrolla la acción.

2.2. Estado del arte reconocimiento de acciones

Durante el tiempo que se lleva investigando esta materia han surgido diferentes puntos de aproximación. Tal y como nos cuenta Rodriguez et al.(50), recopilación que ha servido

como guía a la hora de profundizar en este campo, estos enfoques pueden dividirse en tres grupos principales:

- Técnicas basadas en información de profundidad.
- Métodos basados en extracción del movimiento de forma manual.
- Métodos que usan aprendizaje profundo.

De los tres enfoques, el que se ha considerado de interés ha sido el de aprendizaje profundo, debido al futuro de las tecnologías en este ámbito y a mi propio interés, de forma que será el más detallado de las aproximaciones anteriores.

2.2.1. Técnicas basadas en información de profundidad

A raíz de los avances en tecnologías capaces de capturar la profundidad de una imagen(tal y como lo hacen nuestros propios ojos), se puede obtener mucha más información que la que se extrae con cámaras tradicionales a color, que sólo capturan una imagen RGB individual. Gracias a que los mapas de profundidad que se pueden extraer de las imágenes no se ven afectados por las condiciones de iluminación del ambiente o las sombras, se pueden estudiar más fondo sus características.

En 2012 Yang et al (67) proponen por primera vez el uso de mapas de profundidad, generando los denominados mapas de profundidad del movimiento o Depth Motion Maps (DMM), permitiéndoles extraer las acciones mediante el mapeado de los movimientos realizados. Haciendo uso de histogramas orientados a gradiente o HOG (20), figura 2.1, desde multiples perspectivas ortogonales, fueron capaces de caracterizar la apariencia y forma en los DMM. Esta tecnología pasó a denominarse DMM-HOG, y sus descriptores resultantes son enviados a una máquina de vectores soporte (Support Vector Machines, SVM), que realiza el reconocimiento.

Figura 2.1: Aplicación de HOG a una imagen(6)

Para poder capturar las complejas distribuciones del movimiento de forma temporal, en 2013 Oreifej et al (44) crean un histograma orientado a 4 dimensiones (HON4D), describiendo asi la profundidad, las coordenadas espaciales, y el tiempo. De esta forma su sistema es capaz de capturar los cambios de movimiento junto a un contexto temporal.

Cabe mencionar también el trabajo de Satyamurthi et al (55) en 2018, al proponer avances en el ámbito de los DMM, en este caso presentan las MPDMM o Mapas de profundidad

del movimiento con proyecciones multidireccionales. Su propuesta se basa en convertir los vídeos en frames mediante MPDMM, para posteriormente proyectar la información 3D de profundidad del vídeo en un conjunto de mapas 2D de acuerdo al conjunto de planos y direcciones capturados. Una vez concatenadas estas secuencias de movimiento obtienen el modelo de MPDMM, del cual obtienen posteriormente patrones de textura binaria (Local Binary Pattron), para finalmente aplicar el reconocimiento mediante una ELM (Extreme learning machine) (31).

2.2.2. Técnicas basadas en extracción manual del movimiento

Existen diversas maneras de obtener los atributos representativos de un vídeo, tanto los atributos estáticos de imagen, como los visuales temporales, que son una combinación de los anteriores junto con la información temporal. En el caso que nos concierne son aquellas tecnologías con enfoques basados en el movimiento las que resultan de gran interés.

En 2004 Schuldt et al (56) demostraron que las mediciones de puntos de interés espacio temporales son de gran utilidad para reconocer acciones complejas. Para representar estos patrones, hacen uso de los atributos espacio temporales, que detectan gracias a la aplicación de convoluciones gausianas. Finalmente al igual que en otros métodos de reconocimiento de patrones, hacen uso de un clasificador SVM.

Algunos años después, en 2008, Chen et al (18), presentan un método de reconocimiento de acciones humanas haciendo uso de HOG (20), un método que había demostrado ya eficacia en este campo. En añadido hacen uso de los denominados histogramas orientados a flujo óptico o HOOF, para destacar el movimiento. Hacen uso de una SVM multiclase que es entrenada para clasificar las acciones.

Chaudhry et al (17) en 2009, inspirados por el éxito de la aplicación de histogramas al reconocimiento de acciones en vídeo, deciden optar por aplicar HOOF a cada frame del vídeo. Estos histogramas son creados a partir de computar el flujo óptico a cada frame.

En 2011 Lertniphonphan et al (38), introducen un descriptor del movimiento basado en este caso, en la dirección del flujo óptico.Para el calculo del flujo óptico hacen uso del algoritmo de Lucas-Kanade (41), de tipo disperso.Para finalizar realizan la clasificación mediante agrupamiento de tipo K-means (42).

Algunos años después, en 2016 es destacable la investigación de Kumar et al (37), en la que se introduce la aplicación de descriptores de flujo óptico. Para ello extrae la información del primer plano mediante un algoritmo basado en el modelo de mezcla Gaussiana y técnicas de extracción de flujo óptico. Esta información extraída es usada para alimentar una SVM multiclase.

Recientemente, en 2018, Sehgal et al (57), aplica una técnica de sustracción de fondos a los frames de los vídeos, seguido de una aplicación de HOG. Con la primera aísla las áreas de interés y con el HOG es capaz de describir el movimiento humano. Con esta información entrena una red neuronal con propagación hacia atrás.

2.2.3. Técnicas basadas en aprendizaje profundo

A raíz del éxito cosechado por la aplicación de técnicas de aprendizaje profundo en el campo de la clasificación de imágenes, no era de extrañar que alguien propusiera su utilización para labores de reconocimiento de acciones en vídeos. No obstante, los avances en clasificación de vídeos se han producido mas lentamente, sin haberse conseguido aún precisiones tan altas como en imágenes.

Debido a los notables resultados de la aplicación de redes convoluciones en clasificación de imágenes, Karpathy et al.(33) publicaron en 2014 un estudio sobre las diferentes maneras de extender la conectividad de una red convolucional en un dominio de tiempo, pieza fundamental a la hora de aprovechar la información espacio-temporal procedente de un vídeo. En este estudio se proponen tres variantes de conectividad: fusión temprana, fusión tardía y fusión lenta, figura 2.2. La fusión temprana combina la información en una ventana de tiempo completo, inmediatamente en el nivel de los píxeles. La fusión tardía hace uso de dos redes separadas, que comparten parámetros durante 15 frames, posteriormente los dos flujos son mezclados en la primera capa totalmente conectada. Por ultimo, la fusión lenta, que concebida como un punto intermedio entre los enfoques anteriores, une lentamente la información temporal a lo largo de la red, de tal manera que las capas superiores tienen acceso a información progresivamente más global, tanto en la dimensión espacial como en la temporal.

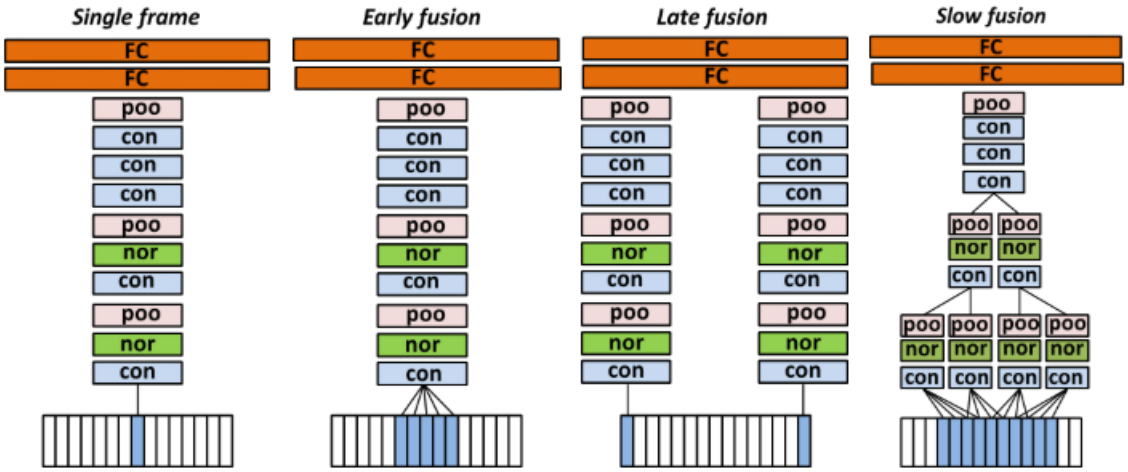

Figura 2.2: Comparativa entre la arquitectura single-frame y las variantes exploradas por Karpathy et al.(33)

A la conclusión de su investigación, comprobaron que el modelo de fusión lenta había sido capaz de obtener las mejores prestaciones de entre los tres propuestos. No obstante, también remarcaron que el modelo "single-frame", o de marco único, había obtenido buenos resultados, sugiriendo que el movimiento local en el vídeo puede no ser tan importante como se creía.

También durante el año 2014, se publicaba la investigación de Simonyan et al.(59), publicación que serviría de precedente para muchos otro estudios posteriores. En esta investigación, Simonyan et al. proponen el uso de una arquitectura de dos flujos de redes convolucionales, de esta manera los vídeos pueden ser divididos en sus componentes espaciales y temporales. Por un lado, la sección espacial, que proporciona información a cerca de la escena y los elementos del vídeo, tomando la entrada mediante un modelo "single-frame".

Por otro lado, la sección temporal, que toma su entrada del resultado de concatenar los vectores de movimiento de la cámara y los elementos de la escena, resultantes de calcular el flujo óptico.

Teniendo en cuenta lo anterior, Simonyan et al. proponen una arquitectura de dos flujos, representada en la figura 2.3. Cada uno de estos flujos es implementado haciendo uso de una red convolucional profunda, cuyos resultados de la capa "softmax", son combinados el uno con el otro mediante fusión tardía.

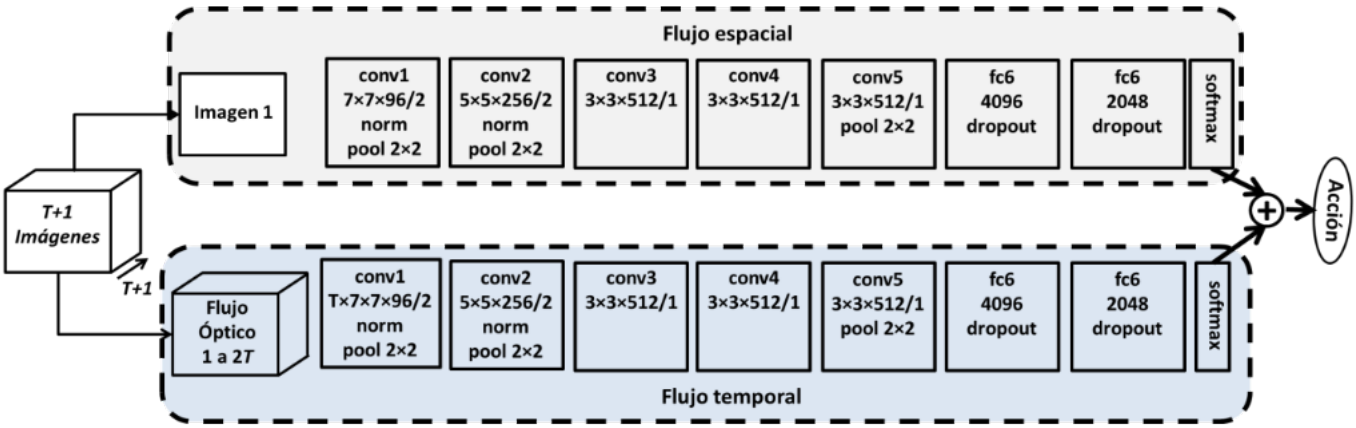

Figura 2.3: Arquitectura de dos flujos para clasificación de vídeos

Dados los resultados de su investigación, concluyeron que entrenar una red convolucional haciendo uso de los vectores de flujo óptico obtenidos del vídeo mejora significativamente los resultados con respecto a hacerlo de la forma convencional.

Tanto el trabajo de Karpathy et al. como el de Simonyan et al. marcaron dos importantes hitos en el ámbito de la clasificación de vídeos mediante aprendizaje profundo, tomándose como punto de partida en algunas de las investigaciones posteriores.

En 2015, Wang et al.(65) presentaron en su trabajo una arquitectura de dos flujos con redes convoluciones de gran profundidad, con el objetivo de mejorar los resultados de la arquitectura anterior. Probaron con las conocidas arquitecturas Googlenet (62) y VGGNet-16 (60), pero en este caso, la red convolucional encargada de la información temporal, utilizaba el flujo óptico correspondiente a secciones de 10 frames continuados del vídeo, calculado mediante el algoritmo TVL1 (68), mientras que para la red espacial seguían usando una única imagen.

Dado el escaso número de ejemplos del dataset usado en su investigación, trabajaron con pesos de los modelos pre-entrenados en el dataset Imagenet (54), como inicialización para las redes. Llegaron a la conclusión de que este pre-entrenamiento no solo había mejorado las métricas en la red espacial, típicamente RGB, sino también de la red temporal, la cual recibía vectores de flujo óptico, que son diferentes.

A largo de este mismo año, Du Tran et al.(63), basándose en el trabajo de Karpathy et al.(33), propusieron el uso de redes convolucionales de tres dimensiones, en lugar de las de dos dimensiones anteriormente utilizadas. Persiguieron como objetivo propagar información temporal a través de todas las capas de la red. Este simple enfoque resultó ser muy efectivo, para el aprendizaje de características espacio-temporales en conjuntos de datos de vídeo supervisado a gran escala. En conclusión, demostraron que las redes convolucionales 3D, junto a un clasificador lineal simple, como una SVM, resultan mas adecuadas para este

tipo de aprendizaje que las redes convolucionales 2D.

Otra interesante parte de la investigación anterior, es que llegaron a usar capas deconvolucionales para interpretar las decisiones de la red. Gracias a estas capas, descubrieron que la red se centraba en la apariencia espacial durante los primeros frames y posteriormente pasaba a fijarse en el movimiento durante los siguientes frames.

En el trabajo publicado en 2016 por Wang et al.(66), se mejoró la arquitectura utilizada en sus investigaciones anteriores, mediante la denominada Temporal Segment Network (TSN) o red de segmento temporal, figura 2.4. La mayoría de los trabajos anteriores no habían sido capaces de incorporar estructuras temporales de largo alcance, sin embargo, Wang et al. consiguieron que su modelo combinara una estrategia de muestreo temporal dispersa y una supervisión a nivel de vídeo detallada, que permitiera un aprendizaje mas eficiente y efectivo, al utilizarse la acción desarrollada durante todo el vídeo. En resumen, hubo dos grandes diferencias con respecto a su investigación previa:

- Muestreo de segmentos del vídeo de forma dispersa, en lugar de aleatoria, mejorando el entendimiento a largo alcance del modelo.
- Nuevas estrategias para predicción a nivel de vídeo:
  1. Combinación de los resultados de los flujos temporal y espacial, de forma independiente, a través de sus fragmentos.
  2. Fusión de los resultados finales de los flujos temporal y espacial, mediante promedio ponderado aplicando *softmax* en todas las clases.

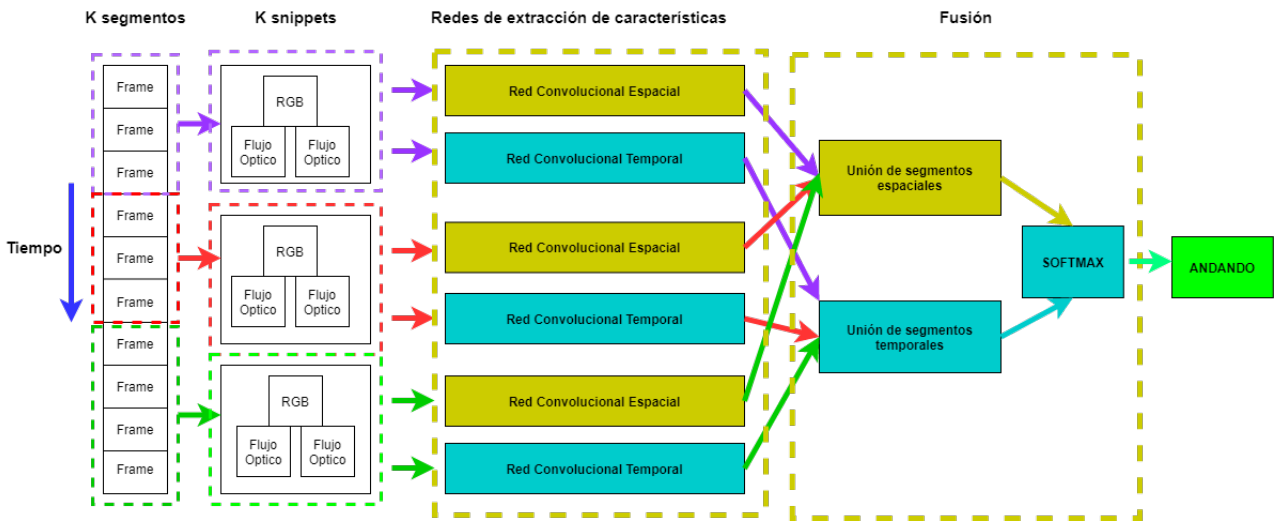

Figura 2.4: Red de segmento temporal o TSN.

En 2017, Carreira y Zisserman, presentaron su investigacion "Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset"(16), en ella proponen un nuevo tipo de arquitectura, que utiliza dos redes diferentes 3D, una para cada flujo dentro de la ya reconocida arquitectura de dos flujos. Esta arquitectura se denomino i3D, ya que se componía de dos flujos de redes convolucionales infladas a tres dimensiones.

Estas redes i3D están basadas en la inflación de las redes convolucionales 2D, expandiendo los filtros y los núcleos de clasificación de las redes convolucionales a 3D, lo que

permite obtener unos clasificadores espacio-temporales de mayor profundidad, permitiendo aprender y extraer mejor las características espacio-temporales de los vídeos.

A diferencia de las tradicionales arquitecturas de dos flujos, en las cuales el flujo correspondiente a la información espacial del vídeo sigue un modelo "single-frame", en las redes i3D, el flujo espacial está compuesto de frames del vídeo apilados en base al tiempo.

Este nuevo modelo se implementó usando como red base la denominada red Inception V1 o GoogleNet (62), pre-entrenada con el dataset Imagenet (54). Las redes Inception V1, son redes convolucionales desarrolladas por Google para hacer frente a tareas de clasificación de imágenes.

Inception V1 se compone de 9 módulos Inception y 2 clasificadores auxiliares. Cada uno de estos módulos Inception aplica una convolución a la entrada con 3 tipos de tamaño diferente (1x1,3x3,5x5). A continuación aplica una capa de max pooling a la entrada, de forma paralela a las convoluciones mencionadas, para finalmente concatenar los resultados y enviarlos al siguiente módulo inception, figura 2.5.

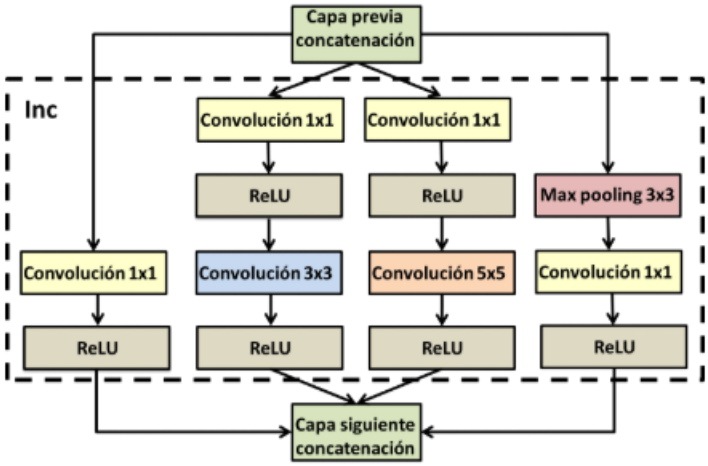

Figura 2.5: Modulo Inception en su versión nativa

Dejando a un lado el nuevo modelo propuesto en este trabajo, otra de las principales contribuciones de la investigación, es la creación de un nuevo conjunto de datos, el *dataset* Kinetics 400 (34), enfocado en acciones humanas. Entre el tipo de acciones que cubre podemos encontrar:

1. Acciones individuales: p. ej. dibujar, beber, reír, puñetazos.
2. Acciones persona-persona: p. ej. dar abrazos, un beso, dar la mano.
3. Acciones persona-objeto: p. ej. abrir regalos, cortar el césped, lavar platos.

Entre sus características podemos destacar que esta compuesto por 400 tipos diferentes de acciones, garantizando como mínimo 400 o mas vídeos únicos por acción. En total el *dataset* tiene unos 240k vídeos de entrenamiento, con duraciones de entre 10-12 segundos cada uno. Posteriormente se han creado los datasets Kinetics-600 (14) y Kinetics 700 (15), siendo cada uno de ellos variantes mas completas del *dataset* original.

Ulla et al.(64) presentaron en 2018 un método de reconocimiento de acciones en vídeos, combinando el uso de redes neuronales convolucionales y redes bidireccionales LSTM (DB-

LSTM). Con idea de reducir la complejidad de computación, lo que hicieron fue extraer los atributos del vídeo en fragmentos de 6 frames, usando una red AlexNet(35) pre-entrenada. La información secuencial extraída es utilizada para entrenar una red DB-LSTM, en la cual multiples frames son concatenados tanto en el paso *forward* como en el paso *backward*. El vídeo es analizado y dividido en secciones o *chunks*, en base a un intervalo de tiempo dado. Siendo el resultado final una combinación de las salidas resultantes de cada *chunk*. Gracias a este tipo de intervalo se consigue información temporal de mayor alcance, permitiendo un mayor entendimiento del vídeo.

También en 2018, destaca el importante trabajo de Hara et al.(25), donde los autores exploran cómo las arquitecturas 2D de vanguardia existentes (como ResNet(27), DenseNet(30), etc.) pueden extenderse a la clasificación de vídeo a través de núcleos 3D. Este cambio, a simple vista sencillo, permite conseguir resultados similares a los alcanzados por la que obtiene mejores resultados hasta el momento, figura 2.6.

De la misma forma que el uso de redes convolucionales 2D pre-entrenadas en el *dataset* Imagenet ha provocado un avance muy significativo en tareas relacionadas con imagen, Hara et al. sacan como conclusión que el uso de redes convolucionales 3D de gran profundidad, pre-entrenadas con el *dataset* Kinetics, podría llegar a alcanzar el mismo éxito de las redes 2D e ImageNet, estimulando así los avances en la visión computarizada dedicada a vídeos.

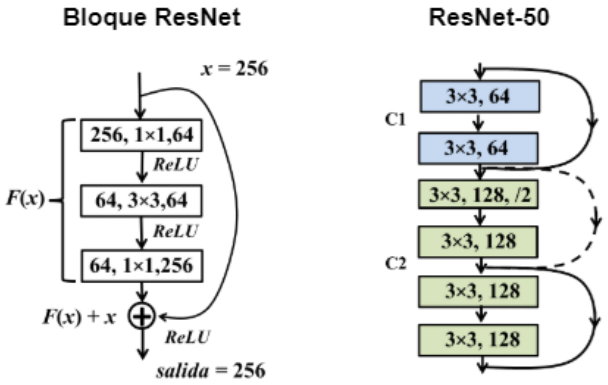

Figura 2.6: Algunos de los bloques de arquitecturas probadas por Hara et al.(25)

2.3. Estado del arte en detección de objetos

Dentro de las ramas de la visión por computador, la detección de objetos en imágenes es una de las más importantes. La posibilidad de detección de objetos en imágenes supone un fuerte apoyo en tareas de estimación de actividades humanas, detección de personas y vehículos o videovigilancia, todas ellas aplicadas en este trabajo.

Pese a sus similitudes, la detección de objetos presenta grandes diferencias con respecto a la clasificación de imágenes. Un algoritmo de detección de objetos en imágenes, busca destacar por medio de cajas delimitadoras las zonas de la imagen en las que ha detectado un cierto objeto con la suficiente confianza.

Uno de los problemas a los que se enfrenta, es que en una misma imagen pueden surgir numerosas detecciones, y no se puede determinar cuántas de antemano. Esto significa, que el tamaño de la capa de salida de la red neuronal es variable.

Una de las primeras formas de solucionar esto consiste en dividir la imagen en determinadas zonas de interés, y aplicar sobre cada una de ellas una red neuronal, de esta forma, es posible determinar la presencia de un objeto en esa región. Sin embargo, uno de los principales problemas que esta primera aproximación saca a relucir, es que no todos los objetos tendrían por qué estar dentro de una de estas regiones, ya que podrían situarse en diferentes posiciones espaciales en la imagen y tener tamaños variables. Por lo tanto, el coste computacional de tener que aplicar una red neuronal a cada una de las regiones posibles dentro de una imagen, teniendo en cuenta lo anterior, sería enorme, de forma que no resulta realizable en la practica.

Para poder detectar y seleccionar las regiones de interés dentro de la imagen, sin tener que someterse a esos costes computacionales prohibitivos, se han desarrollado algoritmos como los explicados a continuación.

2.3.1. R-CNN

Para hacer frente al problema de seleccionar un gran numero de regiones de una imagen, Ross Girshick et al.(24), proponen R-CNN, que se centra en la utilización de una técnica de detección basada en propuestas de región, con el objetivo de detectar objetos en imágenes.

Mediante esta técnica, es capaz de generar un total de 2000 cajas delimitadoras o "bounding boxes", que son enviadas posteriormente a una red neuronal convolucional, que se encarga de extraer las características de estos cajas delimitadoras.

Después de eso, el sistema usa una máquina de vectores de soporte (SVM)(19), que es la encargada en este caso de clasificar si el objeto propuesto es en efecto un objeto o no. En caso de tratarse de un objeto, la SVM también será la encargada de decidir de que objeto se trata. Finalmente es aplicada una función de "supresión no máxima", si se detectó algún objeto más de una vez. Todos las detecciones repetidas de un mismo objeto se eliminan, mediante la anteriormente denominada supresión no máxima..

Éste es un modelo bastante complejo, donde cada uno de esos pasos debe ajustarse específicamente para que funcione correctamente, lo que resulta en una velocidad de rendimiento de cerca de 45 segundos por fotograma, que es considerablemente inferior a otros detectores de objetos, haciendo muy inviable su aplicación en tiempo real.

2.3.2. Fast R-CNN

En 2015, un año despues de la publicacion de R-CNN, Ross Girshick presentó Fast R-CNN (23). Pese al gran éxito que tuvo R-CNN, tenía ciertos problemas que necesitaban ser resueltos, entre ellos:

- Era bastante lento como se ha indicado previamente, ya que necesitaba hacer los cálculos para cada región propuesta.
- Difícil de entrenar. Recordemos que R-CNN debía entrenar 3 partes de forma independiente (CNN, SVM, y el regresor de las cajas delimitadoras)
- Gran gasto de memoria, dado que se necesita almacenar cada resultado de calcular el mapa de información en cada región propuesta.

La manera en que Girshick resolvió estos problemas fue desarrollando esta nueva variante. Una definición simplificada del cambio que supone Fast-RCNN respecto a R-CNN, es que en él se combinan las tres partes diferenciadas presentes en el sistema R-CNN (CNN, SVM, y el regresor de las cajas delimitadoras) en una única arquitectura, figura 2.7.

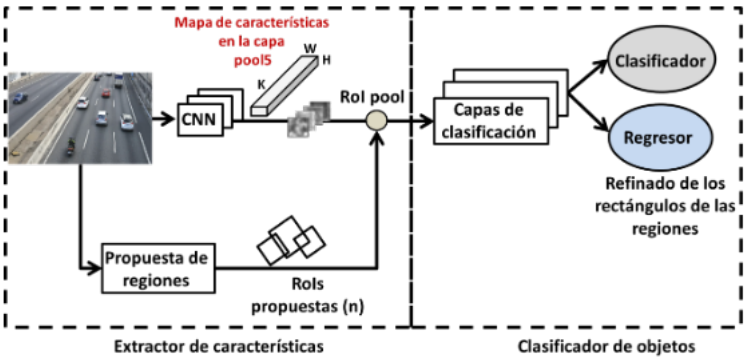

Figura 2.7: Arquitectura Fast-RCNN.

La forma en la que el algoritmo Fast-RCNN funciona se divide en 4 partes:

1. Procesamiento de toda la imagen mediante la CNN, obteniendo el mapa de información de la misma.
2. Para cada una de las regiones propuestas, extrae la parte correspondiente del mapa de información obtenido anteriormente. Seguidamente re-escala el mapa de información de las regiones propuestas, mediante la ayuda de una capa de *pooling* o agrupamiento.
3. El mapa de características de las regiones propuestas es transformado en un vector de información, siempre del mismo tamaño.
4. Se usa el vector obtenido, como entrada para la ultima parte. Debe pasar por una capa *softmax* que decide qué tipos de objetos han sido predichos, y finalmente pasa al regresor de cajas delimitadoras, que da como salida las cajas resultantes para delimitar cada uno de los objetos detectados.

2.3.3. Faster R-CNN

Los algoritmos anteriores (RCNN y Fast-RCNN), hacen uso de un tipo de búsqueda selectiva a la hora de encontrar propuestas de regiones, consumiendo gran cantidad de tiempo. Sin embargo, Faster-RCNN, creado por Shaoqing Ren et al.(49), deja de lado este sistema de búsqueda de regiones, permitiendo que sea la misma red neuronal la que aprenda las propuestas de regiones.

Al igual que Fast-RCNN, la imagen se proporciona como una entrada a una red convolucional que genera un mapa de información como resultado. No obstante, para identificar las propuestas de la región, en lugar de utilizar el algoritmo de búsqueda selectiva en el mapa de información obtenido anteriormente, se utiliza una red separada para predecir las propuestas, la denominada Red de Propuesta de Región (RPN), figura 2.8.

El objetivo de la RPN es generar un conjunto de propuestas, cada una de las cuales tiene una puntuación con su probabilidad de ser un objeto y también el posible tipo de objeto.

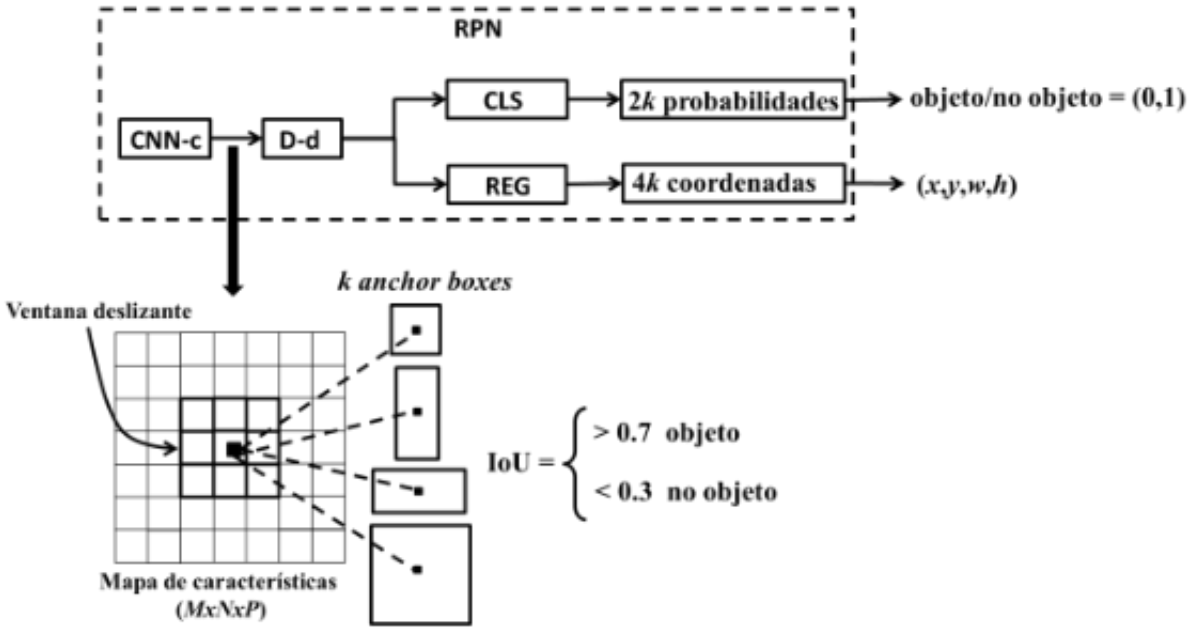

Figura 2.8: Red de propuesta de regiones.

Las propuestas de región predichas por la RPN se vuelven a re-escalar usando una capa de pooling denominada RoI (Region of Interest), que es utilizada para clasificar con ayuda de una R-CNN la imagen dentro de la región propuesta y predecir los valores de desplazamiento para las cajas delimitadoras.

El modelo de red Faster-RCNN es mucho mas rápido que sus predecesores, permitiendo en algunos casos incluso ser usado en tiempo real. Sigue siendo muy ampliamente usado a día de hoy, siendo uno de los mejores algoritmos de detección de objetos disponibles.

2.3.4. YOLO

"You Only Look Once", o YOLO, es uno de los algoritmos de detección de objetos más rápidos que existen, abarcando muchas de las ideas más innovadoras en los colectivos de investigación de visión artificial. Es una muy buena opción cuando se necesita detección en tiempo real, sin conllevar con ello una pérdida de precisión importante.

YOLO es un algoritmo ideado por Redmon et al.(46) en 2015, siendo recibido con gran admiración por los investigadores de visión artificial. El enfoque de este algoritmo es diferente de los anteriores, siendo concebido como una red neuronal capaz de detectar objetos en tiempo real.

YOLO aplica una única red neuronal en la totalidad de la imagen, dividiendo posteriormente la imagen en regiones y predice cajas delimitadores y probabilidades para cada región. Estas cajas delimitadores están ponderados en base a las probabilidades predichas.

Como su propio nombre indica, YOLO solo ”mira una vez” la imagen, en el sentido de que únicamente necesita un paso *forward* en la red neuronal para acabar devolviendo las predicciones. Tras eso, aplica una función de supresión no máxima, para eliminar aquellas detecciones repetidas de un mismo objeto.

Entre las características mas importantes de este algoritmo destacan:

- Su gran velocidad.
- Es capaz de codificar implícitamente información contextual sobre las clases y su apariencia.
- Aprende representaciones generalizadas de los objetos.

Un año después, en 2016, se publicó la siguiente versión del algoritmo, YOLO9000 o YOLO v2. Fue resultado de las investigaciones de Redmon et al.(47), con el objetivo de mejorar la primera implementación. Entre las mejoras destaca la capacidad de detectar 9000 tipos diferentes de objetos.

En 2018, los mismos investigadores publicaron la versión mas modernizada de YOLO, YOLO v3 (48), capaz de ejecutarse significativamente más rápido que otros métodos de detección con un rendimiento comparable. Además, corrige uno de los problemas de las anteriores versiones de YOLO, como es la detección de objetos de pequeño tamaño.

En el caso de YOLO v3, utiliza un red neuronal convolucional denominada DarkNet-53, mas profunda y compleja que la usada en YOLO v2, la DarkNet-19.

En la figura 6.2, se representa la arquitectura detallada de Yolov3.

2.4. Principales tecnologías utilizadas

2.4.1. Python

Python es un lenguaje de programación de alto nivel creado por Guido van Rossum (52), con una gran versatilidad, permitiéndole ser multiplataforma y multiparadigma. Python enfatiza la legibilidad del código y su limpieza, lo que facilita su utilización.

Es un lenguaje interpretado que presenta un fuerte tipado dinámico y una administración de la memoria automática, siendo además, compatible con variados paradigmas de programación, incluidos los imperativos, los orientados a objetos y también, aunque en menor medida, los funcionales.

Python contiene una librería estándar grande y muy completa, pero destaca aún más por la cantidad de librerías y documentación que tiene gracias a grupos de terceros.

En la actualidad uno de los usos más importantes de Python es el análisis de datos, por lo que la gran mayoría de las bibliotecas citadas anteriormente van dirigidas a ser usadas tanto en aprendizaje automático, como en la ciencia de datos, siendo ésta la razón principal por la que se justifica su utilización en este proyecto.

2.4.2. Pytorch

Pytorch (45) es una biblioteca de código abierto con base en la arquitectura Torch, tratándose así de un paquete de computación científica basado en python, con especial hincapié en el uso de unidades de procesamiento gráfico(GPU).

Desde que esta librería fue lanzada en 2016 por Facebook, muchos investigadores han optado por usarla debido a que proporciona la posibilidad de realizar cálculos complejos de tensores con un gran soporte de aceleración por GPU y a su facilidad para cimentar redes neuronales profundas de enorme complejidad.

Al tratarse de una librería muy imperativa crea una metodología ideal para encajar con un estilo de programación *phytonico*, lo que gracias a su interfaz simple, y a sus gráficos computacionales. Además permite una gran flexibilidad a la hora de ser usado en una amplia diversidad de algoritmos diferentes y por esta razón, ha sido utilizado en muchos campos de la computación, como reconocimiento por voz, visión computacional, procesamiento del lenguaje, etc.

Aunque aún no ha conseguido ser adoptado por las masas al nivel que lo es Tensorflow (12) debido a que aún tiene partes en construcción, sí que está generando una dura competencia. La comunidad de Pytorch está creciendo constantemente y está pasando a ser uno de los paquetes principales de todo investigador, siendo usada por gigantes tecnológicos como Facebook, Twitter o Nvidia.

2.4.3. OpenCV

Se trata de una conocida librería de código abierto destinada a labores de visión computacional, desarrollada por Intel y lanzada en el año 2000 (3). Fue construida para proporcionar un entorno de desarrollo sencillo y fácil de utilizar a la hora de crear una infraestructura común para aplicaciones de visión por computador.

La librería permite crear y trabajar con una gran cantidad de algoritmos de visión artificial y de aprendizaje profundo, que pueden ser utilizados para reconocimiento facial, rastrear y detectar objetos, edición de imágenes y de vídeo y otros muchos usos. En particular proporciona los medios para obtener flujos ópticos de diferentes *frames* dentro de un vídeo, permitiendo así procesar los vídeos de la forma prevista.

Grandes compañías hacen uso de esta herramienta a la hora de crear sistemas de detección por videovigilancia o software de identificación de objetos para robots.

Proporciona interfaces en C++, Java, Matlab y Python, razón por la cual, resulta de gran utilidad para el proyecto a la hora de procesar los vídeos.

Figura 2.9: *Frame* resultante de obtener el flujo Fannerback(21) de un vídeo de la plataforma Pexels(6)

.

2.4.4. Pandas

Pandas (5) es una librería de código libre desarrollada por Wes McKinney, fue diseñadas para funcionar como una extensión de Numpy(2) facilitando de gran manera los trabajos a la hora de manipular grandes conjuntos de datos.

Este paquete para Python proporciona estructuras de datos rápidas, flexibles y diseñadas para ser intuitivas y fáciles de usar.

Actualmente el equipo que mantiene esta librería tiene el objetivo de convertirla en la herramienta de análisis y tratamiento de datos más potente de código libre, y ya están en camino hacia ese objetivo.

Algunas de sus funcionalidades más destacables son:

- Mutabilidad de tamaño: las columnas se pueden insertar y eliminar del DataFrame.
- Reestructuración y segmentación de conjuntos de datos.
- Conjuntos intuitivos de unión de datos.
- Herramientas de entrada/salida robustas que permiten cargar datos desde archivos planos(CSV), archivos de Excel, bases de datos, etc.
- Alineación de datos de forma automática.

```
1  # Import cars data
2  import pandas as pd
3  cars = pd.read_csv('cars.csv', index_col = 0)
4
5  # Print out observation for Japan
6  print(cars.iloc[2])
7
8  # Print out observations for Australia and Egypt
9  print(cars.loc[['AUS', 'EG']])
```

Figura 2.10: Ejemplo de carga de dataframe a partir de un .csv

2.4.5. Numpy

Numpy (2) es una librería de código abierto para Python que le aporta mayor soporte a la hora de realizar cálculos científicos con vectores y matrices.

Fue creada por Travis Oliphant en 2005, basándose en Numeric, una librería anterior creada por Jim Hugunin diez años antes. Oliphant desarrolló un código más fácil de mantener y lo suficientemente óptimo y flexible para implementar las características novedosas de manejo de arrays que implementa Numarray.

La importancia de Numpy ante todo es su capacidad de proporcionar estructuras para encapsular diferentes tipos de datos implementados de manera propia, como arrays o matrices, las cuales son mucho más rápidas y eficientes que las proporcionadas de forma estándar por Python.

Capítulo 3

Flujo Óptico

3.1. Introducción

En su variante biológica, la capacidad de detectar el movimiento mediante el sentido de la vista, es una característica esencial, tanto en los humanos, como en los animales. Sin embargo, esta capacidad típicamente biológica, puede ser replicada de forma artificial, siendo en este caso una cámara de vídeo, la que capturará la información, y no la vista.

Cuando hablamos de vídeos, debemos comprender que éstos se componen de varias imágenes concatenadas una detrás de otra, que mostradas a una determinada velocidad dan la apariencia de continuidad. En base a esto, debemos entender que una secuencia de imágenes es aquella en la que participan dos o mas imágenes de una determinada escena.

Las imágenes que componen una secuencia son diferentes entre sí, dado que pertenecen a diferentes momentos en el tiempo; no obstante, cuando en una escena hay ciertos elementos en movimiento, descubriremos que se genera una variación espacio-temporal que relaciona las imágenes entre sí, figura 3.1. Esta variación espacio-temporal suele ser de carácter suave, en la medida en que no se hayan producido cambios drásticos a lo largo de la escena, y la diferencia temporal entre las imágenes sea pequeña.

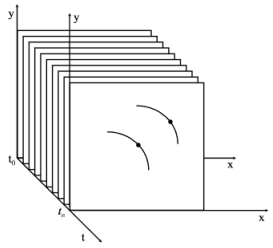

Figura 3.1: Representación espacio-temporal del movimiento

Normalmente el análisis del movimiento en un vídeo, se basa en la concatenación de un cierto numero de imágenes consecutivas, algunas veces dos o tres en una secuencia. La idea en la que se focaliza este análisis de vídeos, es la de determinar el movimiento existente mediante la búsqueda de similitudes entre ciertos puntos de interés en las imágenes que componen la secuencia.

3.2. Estimación del flujo óptico

El concepto de flujo óptico no es algo nuevo, ya durante los años cuarenta, el psicólogo estadounidense J.Gibson, publicó un estudio (22) sobre las observaciones realizadas a las diferentes fotografías que tomó de un avión en movimiento.

El flujo óptico permite ver reflejados los cambios en una imagen como consecuencia de un movimiento durante un intervalo de tiempo dt. Permitiendo además, determinar tanto la dirección del movimiento que se ha llevado a cabo, como la velocidad de dicho movimiento. No obstante, como se ha mencionado previamente, se requiere que el intervalo temporal que separa dos imágenes consecutivas sea suficientemente pequeño, y que no ocurran cambios muy notables entre ellas. A la hora de obtener el flujo óptico, debemos tener en cuenta dos restricciones:

- La intensidad observada para cualquier punto de un objeto es constante en el tiempo, bajo las condiciones específicas de movimiento de las cámaras o movimiento de los objetos en la escena.
- Los píxeles que rodean al observado tendrán un movimiento similar.

Si tomamos en cuenta la función de intensidad $f(x, y, t)$, de un píxel localizado en la posición espacial (x, y), en un instante t, podemos definir la ecuación de flujo óptico como:

$$\frac{\partial f}{\partial t} + \mathbf{v}.\mathbf{G} = 0 \tag{3.1}$$

Siendo $\mathbf{v}$ el vector de velocidad y $\mathbf{G}$ el equivalente a $grad(f)$, es decir, el gradiente espacial de una imagen en la localización (x, y).

Si llevamos a cabo un desarrollo en serie de Taylor de f obtenemos la expresion:

$$f(x + dx, y + dy, t + dt) = f(x, y, t) + f_x dx + f_y dy + f_t dt + O(\partial^2) \tag{3.2}$$

donde $O(\partial^2)$ se consideran términos de orden superior despreciables.

Considerando que un píxel (x, y, t), se desplaza una pequeña distancia (dx, dy) en el espacio, respecto a la posición en la imagen anterior y con una diferencia temporal dt, podemos expresar:

$$f(x + dx, y + dy, t + dt) = f(x, y, t) \tag{3.3}$$

Tomando en cuenta lo anterior, y despreciando los términos de orden superior de la ecuación resultante del desarrollo en serie de Taylor, la ecuación del flujo óptico quedaría de la siguiente forma:

$$-f_t = f_x \frac{dx}{dt} + f_y \frac{dy}{dt} \tag{3.4}$$

En base a esta ecuación podemos determinar $v = \left(\frac{dx}{dt}, \frac{dy}{dt}\right) = (u, v)$, sabiendo que f_x, f_y y f_t son todas cantidades medibles.

Hay que destacar que la ecuación anterior asume que la intensidad de la imagen permanecerá constante a lo largo de la trayectoria s del movimiento, resultando $df/ds = 0$. Sin embargo la suposición anterior no siempre es cierta, y la intensidad no permanece constante, en especial si hay un cambio de iluminación. Para lidiar con esta posibilidad, se aplica una restricción de gradiente constante a lo largo de s, de forma que aunque la intensidad varíe, el gradiente sí sera constante, siendo $\frac{d\bigtriangledown f}{ds} = 0$.

3.3. Métodos de flujo óptico disperso

Los métodos de flujo óptico disperso hacen una selección de las características mas importantes a nivel de píxel (p.e. las esquinas o bordes de un objeto) y calculan y trazan sus velocidades de desplazamiento. Estas características, una vez extraídas, son utilizadas en el corazón de la función de flujo óptico *frame* a *frame*, de manera que se pueda garantizar que son los mismos puntos los que están siendo desplazados. Entre los métodos de flujo óptico disperso destacan el método Horn-Schunck(29), el método Buxton-Buxton(13) y en especial el reconocido método de Lucas-Kanade(41).

Para poder utilizar un método de flujo óptico disperso, solo necesitamos mantener observados una serie de puntos de interés previamente calculados. Para calcular estos puntos se suele hacer uso de algoritmos de detección de bordes y esquinas, destacando el método de detección de esquinas de Shi-Tomasi(58) y el método de Harris(26).

3.3.1. Método de Lucas-Kanade

La aproximación propuesta por Lucas-Kanade(41) al problema de la detección de movimiento en vídeos está enfocada a la comparación de dos *frames* consecutivos de un vídeo. Cuando se utiliza el método de Lucas-Kanade deben asumirse dos cosas:

- La diferencia temporal entre los dos *frames* debe ser pequeña, de forma que un objeto no sufra un gran desplazamiento de un *frame* a otro.
- El *frame* debe ser representado mediante tonos de gris, con una variación suave de los mismos.

Teniendo en cuenta estas asunciones, y partiendo de la ecuación de estimación de flujo óptico de la sección 4.2, el algoritmo de Lucas-Kanade es representado de la forma:

$$\begin{bmatrix} \partial^2 f/\partial x^2 & \partial^2 f/\partial x \partial y \\ \partial^2 f/\partial x \partial y & \partial^2 f/\partial y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = -\frac{\partial(\vec{\nabla} f)}{\partial t} \tag{3.5}$$

considerando un entorno de vecindad Ω alrededor de cada punto sobre el que se extiende el sumatorio, la ecuación (4.5) se puede expresar como sigue,

$$\begin{bmatrix} \sum_\Omega f_x^2 & \sum_\Omega f_x f_y \\ \sum_\Omega f_x f_y & \sum_\Omega f_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = -\begin{bmatrix} \sum_\Omega f_t f_x \\ \sum_\Omega f_t f_y \end{bmatrix} = A\mathbf{v} = \mathbf{b} \tag{3.6}$$

Si se despeja $\mathbf{v}$ de la ecuación 4.6 quedaría de la forma:

$$\mathbf{v} = (A^T A)^{-1} A^T b \tag{3.7}$$

3.4. Métodos de flujo óptico denso

A diferencia del flujo óptico disperso, donde el calculo del flujo óptico se aplica únicamente a unos puntos calculados previamente, los algoritmos de flujo óptico denso calculan los vectores de flujo óptico para todos los puntos del frame. Esta práctica presenta resultados mucho mas precisos que la dispersa, permitiendo generar resultados adecuados para labores de aprendizaje y reconocimiento de movimientos. Sin embargo, esta mejora de precisión conlleva un elevado coste de computación en comparación con los métodos dispersos.

De entre los métodos de calculo del flujo óptico denso destacan el método Gunner-Fanerback(21) y el TV-L1(68)], utilizado en tareas de reconocimiento de vídeo. En especial, el método mas investigado y probado durante el desarrollo de este proyecto es TV-L1, sin embargo, como se explica mas adelante no formará parte del sistema final.

3.4.1. Método TV-L1

De entre los distintos enfoques para estimar el flujo óptico, el método TV-L1(68), es ampliamente reconocido dado su gran nivel de equilibrio entre precisión y eficacia.

Una sección de vídeo puede expresarse como una función de la forma $I_t(x, y)$, donde x e y representan las coordenadas espaciales de un punto, y t representa el tiempo. El valor de esa función simboliza el brillo del píxel en la posición $x = (x, y)$. Un punto x puede desplazarse en el vídeo a la largo del tiempo, siendo $u^t(x) = (u_1^t(x), u_2^t(x))$ la forma de representar el desplazamiento de dicho punto del frame t al frame $t + 1$.

Partiendo de dos *frames* consecutivos denominados I_0 yI_1. La meta perseguida es minimizar un error basado en imágenes usando una función de regularización. Como medida para calcular la similitud entre imágenes se utilizan las diferencias de intensidad entre los píxeles. La principal representación de este algoritmo viene dada de la siguiente forma:

$$minimo : u(x), x \in \Omega \sum_{x \in \Omega} (|\nabla u_1| + |\nabla u_2|) + \lambda |p(u(x))|, \tag{3.8}$$

siendo Ω todas las coordenadas de píxeles de la imagen. Donde el término $|\nabla u_1| + |\nabla u_2|$ representa la condición de suavidad del brillo y el término $p(u(x))$ representa la asunción de la constancia de brillo, formulada de la forma $I_0(x + u) \approx I_1(x)$.

La figura 4.2 muestra en su parte izquierda un ejemplo de cómputo del flujo óptico TV-L1 mostrándose la magnitud del movimiento mediante la representación de distintas tonalidades de color, tal y como se muestra en la parte derecha

Figura 3.2: Flujo óptico TV-L1 obtenido de un vídeo (61)

Capítulo 4

Redes Neuronales Artificiales

4.1. Introducción

Si describiéramos una red neuronal de forma concisa, podríamos verla como un canon de la programación con inspiración en la vida, mas concretamente, inspiradas en las conexiones de las neuronas del cerebro, y que nos permite procesar datos a partir de los cuales posibilitar que un sistema aprenda.

Ahora mismo las redes neuronales son una de las vertientes con mayor potencial a la hora de conseguir detectar humanos en vídeos o imágenes, y mas concretamente poder identificar las acciones que éstos realizan. Además, gracias a su gran versatilidad, han demostrado ser capaces de obtener muy buenos resultados en otros campos, como el procesamiento de lenguaje natural, problemas de predicción, etc.

Aunque los modelos matemáticos inspirados en esta rama de la biología se introdujeron en 1943, por McCulloh y Pitts(43), no fue hasta unos años después, en 1950, cuando Rossemblat creó el Perceptrón(51), unidad básica de la que nacerían las redes neuronales. El denominado Perceptrón es un algoritmo binario de clasificación, que produce una salida de 1 o 0. Fue aquí donde por primera vez se introdujo el concepto de "pesos", un número real que permite reflejar la consideración que se le da a una determinada entrada con respecto a la salida.

4.2. La neurona biológica

Desde el punto de vista biológico una red neuronal es en definición un conjunto de más de cien mil millones de neuronas conectadas entre sí, que transmiten información de una a otra mediante una serie de impulsos eléctricos. Si colocáramos una de estas neuronas bajo el microscopio podríamos ser capaces de diferenciar las partes que la componen, y de forma sencilla comprender las similitudes de una neurona real con una neurona artificial. Aunque las neuronas biológicas pueden variar en tamaño y forma, se componen de tres partes esenciales, figura 4.1:

- El **cuerpo o soma** de la neurona, contiene el núcleo celular, que mantiene la estructura de la neurona y proporciona energía para impulsar las actividades.

- Las **dendritas** son raíces fibrosas que se ramifican desde el cuerpo celular. Son las encargadas de recibir y procesar las señales eléctricas provenientes del axón de otra neurona.
- El **axón** es una estructura alargada unida al núcleo de la neurona, por la cual se envían los impulsos eléctricos una vez han salido del núcleo, y solo si estos impulsos superan un cierto valor umbral mínimo.

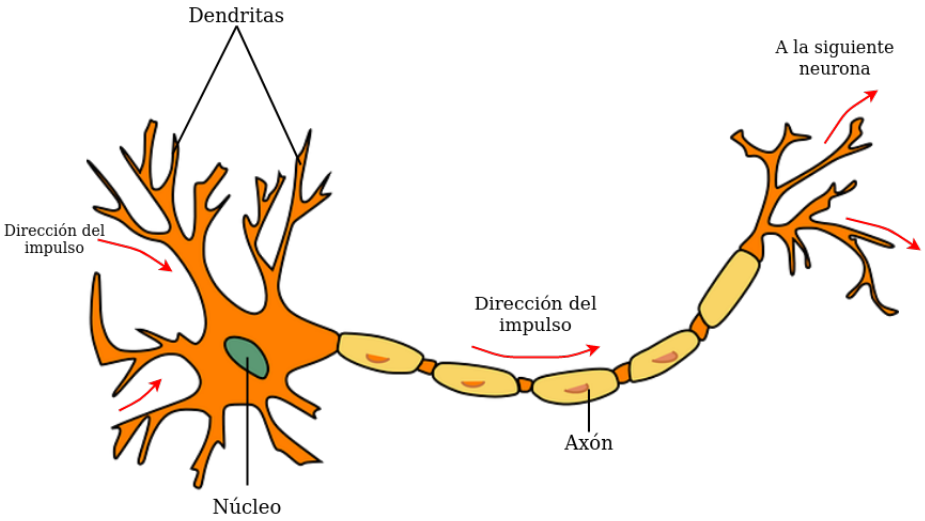

Figura 4.1: Neurona Biológica

Sabiendo lo citado anteriormente y entendiendo las partes que conforman la neurona biológica, podemos ver que existe una gran correlación entre la neurona biológica y la artificial. La neurona artificial consta de una zona de entrada que hace la función de las dendritas en la biológica, tiene una zona central que realiza las operaciones, al igual que la biológica tiene el núcleo, y por último posee una parte de salida paralelamente al axón en la neurona biológica.

4.3. La neurona artificial

Como hemos visto la neurona artificial tiene cierta semejanza con la neurona biológica, no obstante merece la pena mencionar todas las partes que la conforman. Una neurona artificial consta de las siguientes partes:

- Un número n de variables de entrada, siempre cumpliendo n>0. Generalmente se representa en forma de vector de entradas, de forma $x = (x_1, x_2, x_3..x_n)$.
- Un **peso** asignado a cada **variable de entrada**, que define la importancia o "fuerza"de cada conexión. Estos pesos se representan en forma de vector de la forma $w = (w_1, w_2, w_3..w_n)$
- El **sesgo** o bias, b, es un parámetro que indica la viabilidad que tiene una neurona para activarse, de la misma forma que en las neuronas biológicas los impulsos eléctricos deben superar un determinado umbral para ser propagados.

- La **función de propagación**, es la aplicada en las variables de entrada, los pesos y en el sesgo tras la entrada, antes de la función de activación, $y = f(x_1...x_n, w_1...w_n, b)$.

  En su versión mas básica, la perteneciente al perceptrón es de tipo lineal y se representa:

$$y = \sum_{i=1}^{n} w_i x_i + b \tag{4.1}$$

- La **función de activación**, sera la responsable de determinar si se activa o no la salida de la neurona en función del valor resultante de la función de propagación, expresado de la forma $z = f_{act}(y)$. Dos funciones de activación muy utilizadas son la función sigmoide, que produce una salida entre 0 y 1, y la función de unidad lineal rectificada o RELU, ambas representadas respectivamente de la forma:

$$Sigmoide : f_{act}(y) = \frac{1}{1 - e^{-y}} \tag{4.2}$$

,

$$RELU : f_{act}(y) = max(0, y) \tag{4.3}$$

La figura 4.2 muestra un esquema de una neurona artificial.

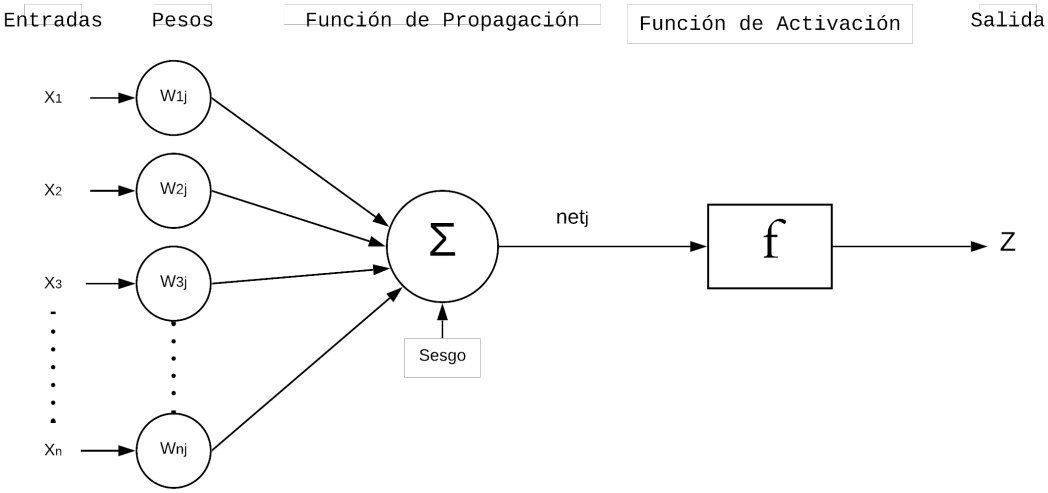

Figura 4.2: Esquema de la Neurona Artificial

4.4. Estructura de las redes neuronales

Si se quiere llevar a cabo cálculos verdaderamente complejos, con una sola neurona no es suficiente, se deberán utilizar muchas de ellas interconectadas entre sí. Por norma general, las neuronas se agrupan o constituyen conjuntos conformando las denominadas redes neuronales.

Entre las estructuras más utilizadas se tienen las redes recurrentes, y especialmente las redes *feed-forward*, que suponen el tipo de red mas elemental. No obstante, otro tipo de red muy utilizada también en el ámbito de detección en imágenes y vídeos, bajo el paradigma

de aprendizaje profundo, son las redes neuronales convolucionales en general y dentro de éstas las redes residuales.

4.4.1. Redes de tipo feed-forward

Las redes *feed-forward* (hacia delante) reflejan la idea elemental que subyace en una red neuronal, las capas. En estas redes, encontramos agrupadas las neuronas en conjuntos denominados capas. Es en el momento en el que conectamos varias de estas capas, cuando hablamos de conformar una red neuronal. Entre estas capas distinguimos tres tipos: la capa de entrada, un número n de capas ocultas, y la capa de salida.

- **Capa de entrada**: esta capa no consta de operaciones matemáticas en su interior, sino que se corresponde con el conjunto de datos de entrada. Tiene tantas neuronas como elementos tiene el vector de entrada.
- **Capas ocultas**: contienen toda la lógica matemática intermedia de la red. Está conformada por neuronas ocultas, denominadas de esa manera porque las salidas de las neuronas de dichas capas se interconectan con otras neuronas, de forma que no pueden ser "vistas", desde fuera de la red.
- **Capa de salida**: es la capa final de la red, la que devuelve la predicción objetivo. Tiene tantas neuronas como número de clases posibles haya en el problema.

En este tipo de red, las neuronas se encuentran conectadas de capa en capa, hasta llegar a la capa de salida. Las neuronas de una capa, únicamente pueden estar conectadas a las neuronas de la capa inmediatamente siguiente.

En función del número de variables de entradas de la red, tendremos mayor o menor número de neuronas, siendo este número igual al de entradas. Por otro lado, el número de datos de salida dependerá de la cantidad de variables necesarias para representar la salida en cuestión. La figura 4.3 muestra un esquema de una red de tipo feed-fordward con sus correspondientes capas, de entrada y salida y dos capas ocultas.

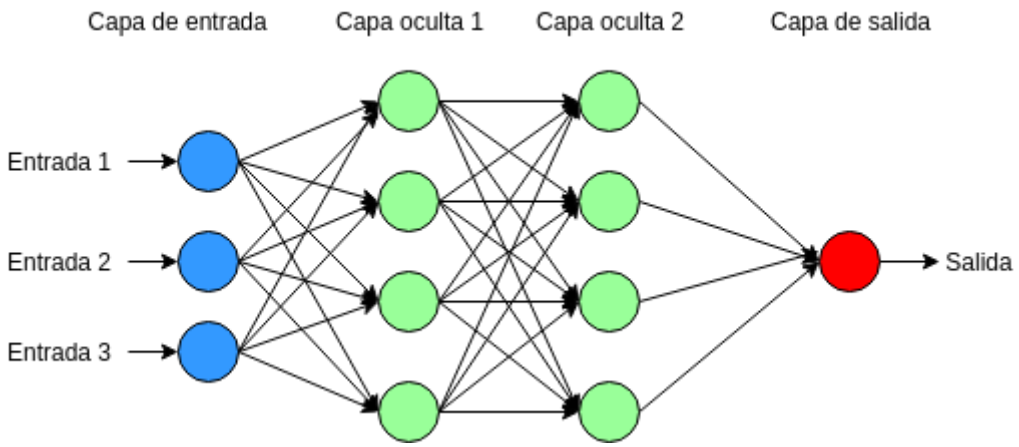

Figura 4.3: Esquema de una red *feed-forward*

Podemos definir la salida de una capa n de red *feed-forward* mediante la siguiente

fórmula matemática:

$$z_i^n = f_{activacin}\left(\sum_{j=1}^{J} z_j^{n-1} w_{ij}^n + b_i^n\right) \tag{4.4}$$

Donde z_j^{n-1} es la salida de la neurona j de la capa n-1 de la red, la cual tiene un total de J neuronas. Por otra parte, tenemos w_{ij}^n, que es el peso de conexión de la neurona i de la capa n, con la neurona j de la capa n-1, y el sesgo b_i^n , de la neurona i de la capa n. Finalmente se define z^n, vector de salida equivalente a todas las posibles z_i^n de la capa n.

Una vez quedan claros estos conceptos principales, definiremos las redes profundas como aquellas redes con un gran número de capas ocultas, siendo éstas las utilizadas dentro del conocido paradigma del aprendizaje profundo.

4.4.2. Redes de tipo recurrente

Uno de los elementos que más caracterizan el razonamiento humano, es la existencia de una memoria, gracias al conocimiento que en ella almacenamos, podemos tomar unas u otras decisiones en el futuro. Por ejemplo, cuando leemos un libro somos capaces de comprender los significados de ciertas expresiones gracias al contexto que forman las palabras anteriores que hemos memorizado. Basándose en esa idea surgieron las redes neuronales recurrentes o RNN (Recurrent Neural Networks) (28), que son una de las arquitecturas mas cercanas al razonamiento humano.

Las redes RNN, al igual que las demás redes, aprenden durante el proceso de entrenamiento, pero además, son capaces de tener en cuenta lo aprendido de entradas anteriores, alterando así la salida. Esta idea es la que otorga a la red el concepto de temporalidad, permitiendo a las redes tener memoria. Precisamente este concepto temporal es el que permite establecer la conexión entre las secuencias de vídeo, que llevan implícita la vertiente temporal, y este tipo de redes.

Esta información adicional que recibe la red proviene del denominado vector de estado oculto o S, que es el que representa ese contexto previo. Una vez generada la salida en función de la entrada y el estado oculto, el vector de estado oculto es actualizado en base a dicha entrada, quedando preparado para usarse en la siguiente entrada, figura 4.4.

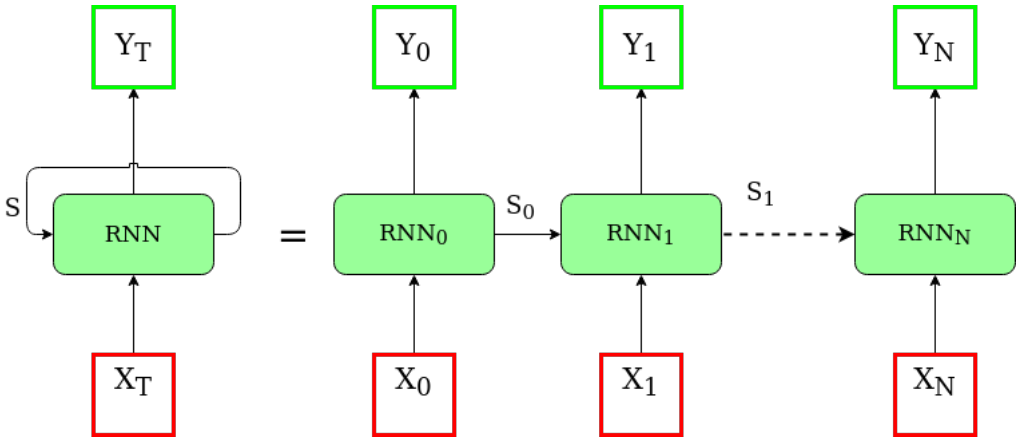

Figura 4.4: Arquitectura básica RNN desenrollada

La figura 5.4 muestra un esquema general de las RNN, constituida por una serie de celdas de memoria, que generan salidas Y_i y además cada celda alimenta a la siguiente en la secuencia temporal.

4.4.3. Redes de tipo residual

Este tipo de estructura de red neuronal surgieron en 2016, cuando He et al.(27) se plantearon la creciente dificultad para entrenar redes profundas. Es precisamente en la búsqueda de una mayor facilidad de entrenamiento o mayor facilidad en el proceso de optimización, donde surge el nuevo paradigma de las redes residuales.

En ciertos casos puede ocurrir que cuando la profundidad de una red neuronal aumenta, suele tender a saturarse la precisión, sin necesariamente ser por culpa del sobreajuste. La idea subyacente detrás de las redes residuales o ResNet, tiene como aporte principal, la introducción de un "cortocircuito", que permiten realizar conexiones que saltan una o más capas de la red.

En lugar de esperar que cada cierto número de capas apiladas, éstas encajen o se acoplen directamente hacia una proyección subyacente deseada, se deja explícitamente que estas capas se ajusten a una proyección residual. Si denominamos a la proyección subyacente deseada de la forma H(x), se permite que las capas no lineales apiladas se ajusten a una proyección F(x) = H(x)-x. De esta manera, la proyección original se planteará de la forma F(x) + x, bajo la hipótesis de que si una proyección de identidad es óptima, resultaría más fácil llevar el residuo F(x) a cero, que ajustar una proyección de identidad mediante una pila de capas no lineales. Es decir, es más fácil encontrar una solución tal que F(x) = 0, en lugar de F(x) = x, usando una pila de capas no lineales como función.

La función F(x) es la que denominamos residuo. Bajo esta formulación se argumenta que las capas apiladas no deberían empeorar el rendimiento de la red, porque simplemente se pueden apilar asignaciones de identidad, es decir, una capa que no hace nada. En cualquier caso, conviene dejar claro que la red posee tanto las capas apiladas no lineales, que producen la salida F(x), como el cortocircuito que produce la identidad x, figura 4.5.

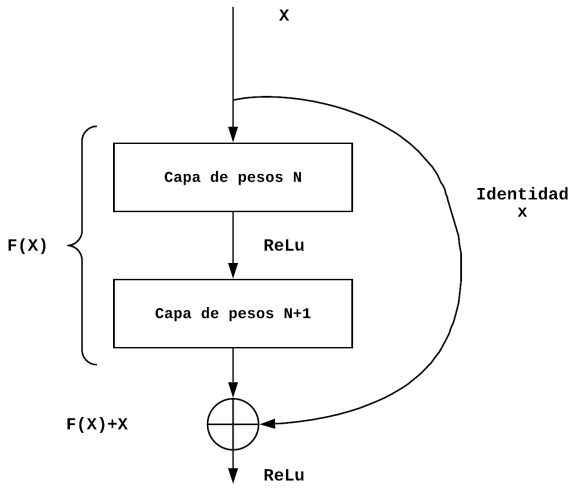

Figura 4.5: Esquema de un bloque de red residual

4.5. Tipos de aprendizaje en las redes neuronales

De igual manera que pueden distinguirse diferentes variantes de red neuronales en base a su estructura, otra forma de clasificarlas es respecto al tipo de aprendizaje que utilicen.

Existen un total de 4 vertientes de aprendizaje con las que un algoritmo de aprendizaje automático puede aprender: **aprendizaje supervisado, aprendizaje no supervisado, aprendizaje semi-supervisado o híbrido y aprendizaje por refuerzo**. De entre los anteriores se pueden definir dos tipos claramente diferentes, aquellos en los que se necesita conocimiento de los datos con antelación y aquel que precisamente se caracterizada por su ausencia.

4.5.1. Aprendizaje supervisado

En las redes neuronales que hacen uso de este tipo de aprendizaje, se parte de un conocimiento previo que debe ser entregado a la red. El objetivo es que mediante un conjunto de datos de entrada y la salida que debería devolver la red, se pueda inferir una función capaz de establecer una proyección de las entradas con las salidas de la manera más precisa posible. Estos datos de entrenamiento, son a menudo denominados tuplas (X,Y), siendo X los datos de entrada e Y los datos de salida o etiquetas. La figura 4.6 muestra un esquema simplificado del aprendizaje supervisado, desde la entrada a la salida pasando por el proceso de entrenamiento e incluyendo los datos de test para validación del modelo de red.

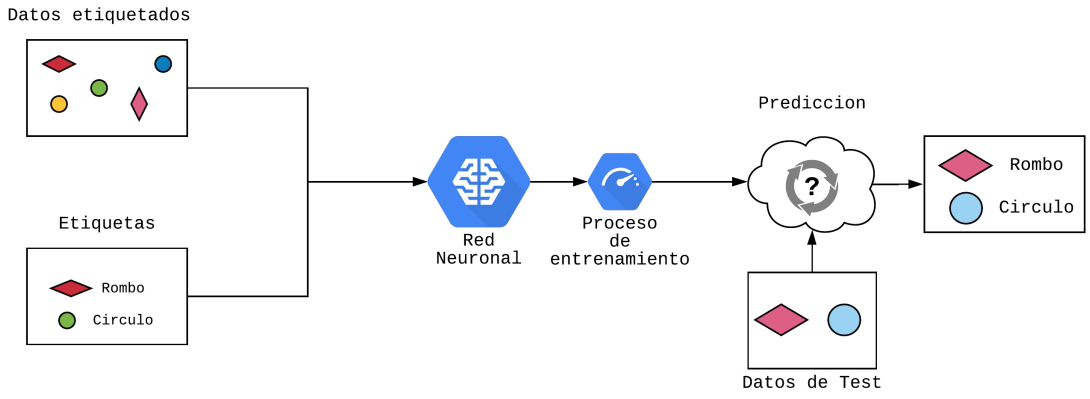

Figura 4.6: Esquema simplificado del aprendizaje supervisado

Durante el proceso de entrenamiento, la red neuronal ajustará sus pesos internos para que la salida de la red sea cada vez mas similar a la salida deseada. Llevando a cabo una serie de iteraciones, hará uso de métodos capaces de calcular el grado de error o pérdida que tiene la red en cada momento, con el objetivo de disminuir dicho error mediante el citado ajuste de los pesos.

4.5.2. Aprendizaje no supervisado

A diferencia del aprendizaje supervisado, en este tipo de aprendizaje se presenta una serie de datos de entrada, pero no los correspondientes datos de salida, es decir, no existe conocimiento previo. No recibe como entradas tuplas (X,Y), sino solo X.

El objetivo del aprendizaje supervisado es el de modelizar y distribuir los datos de manera que permita extraer conclusiones sobre ellos. Permite descubrir estructuras interesantes en los datos y estimar funciones de densidad de probabilidad de aparición de los patrones en el conjunto de datos.

De forma general, los algoritmos que utilizan este tipo de aprendizaje pueden ser agrupados en algoritmos de *clustering* o agrupación y algoritmos de asociación.

4.5.3. Aprendizaje semi-supervisado o híbrido

El aprendizaje de este tipo se encuentra en un punto intermedio entre el aprendizaje supervisado y el no supervisado. Se tienen por un lado, tanto datos etiquetados de la forma de tupla (X,Y) y por otro datos sin etiquetar de la forma X. Generalmente utiliza una pequeña cantidad de datos etiquetados y un gran número de datos no etiquetados.

La razón por la que se lleva a cabo este tipo de aprendizaje es porque ha demostrado en algunos casos mejorar de manera notable la precisión del aprendizaje. No obstante, otro de los puntos a favor de este tipo de aprendizaje es que permite usar pocos ejemplos etiquetados para entrenar. Conseguir una gran cantidad de datos etiquetados para resolver un cierto tipo de tarea suele requerir en ciertos casos de la presencia de un elemento humano que clasifique manualmente los datos. Muchas veces, un proceso de etiquetado de este tipo lleva acarreado un gran coste, llegando a hacer que conseguir ciertos conjuntos de datos de entrenamiento sea inviable, mientras que conseguir datos no etiquetados es relativamente fácil. En el tipo de situaciones en las que se presente un caso así, el aprendizaje semi supervisado podría resultar de gran utilidad.

4.5.4. Aprendizaje por refuerzo

En el caso del aprendizaje por refuerzo, se presenta un escenario, que al igual que el híbrido, está situado en un punto intermedio entre el aprendizaje supervisado y el no supervisado. Sin embargo, en este tipo de aprendizaje no se proporciona a la red la respuesta deseada, sino que se le indica el error que comete de forma global.

Este tipo de aprendizaje esta basado en el conductismo, una rama de la psicología dedicada al estudio de la conducta en base a estímulos y recompensas. Si abstraemos este concepto al aprendizaje automático, descubrimos que lo que se busca definir es el comportamiento que debe tener nuestra red neuronal para maximizar algún tipo de recompensa. La figura 4.7 muestra un esquema de funcionamiento del aprendizaje por refuerzo, donde se muestra cómo la red es recompensada cuando la acción realizada se asume como correcta.

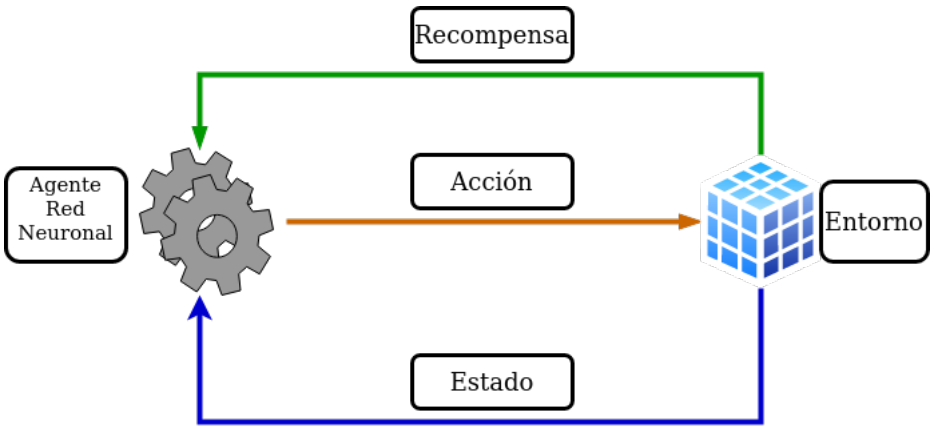

Figura 4.7: Esquema simplificado del aprendizaje reforzado

4.6. Métodos de aprendizaje

La teoría subyacente en las redes neuronales tiene una gran extensión y repercusión, de forma que en este apartado se describen brevemente los métodos de aprendizaje más relevantes utilizados en las redes neuronales y cuyas ideas resultan de gran importancia a la hora de entender el funcionamiento interno de las redes neuronales durante el proceso de entrenamiento. El siguiente paso después de definir la estructura de la red neuronal, es encontrar el valor de los diferentes parámetros que la componen, buscando en todo momento alcanzar las mayores precisiones del modelo de red en cuestión.

4.6.1. Función de coste

Esta función, también conocida como función de pérdida, tiene como objetivo medir cómo de equivocados están siendo los resultados de un modelo de aprendizaje automático. Simplificando su funcionamiento, puede definirse como la comprobación de la diferencia existente entre las respuestas dadas por el modelo, con respecto a las respuestas correctas que debería haber generado. De forma que permite realizar una medición sobre cómo de bien están ajustados los parámetros de dicho modelo con respecto al conjunto de datos de entrenamiento.

A la hora de abordar un problema, resulta prioritario elegir una función de coste adecuada, ya que los parámetros del modelo de redes neuronales serán ajustados con el objetivo de minimizar dicha función.

De entre las funciones de coste mas utilizadas destaca *Mean Squared Error* (MSE) o error cuadrático medio. Simplificada de la forma:

$$MSE = \frac{1}{M}\sum_{i=1}^{M}(x_i - y_i)^2 \tag{4.5}$$

Siendo x_i el valor de salida esperado para una determinada entrada, y_i el valor dado por la red para esa misma entrada y M el número de entradas.

Otra función de coste a destacar es la función *Cross-Entropy* (CE) o entropía-cruzada, usada generalmente en problemas de clasificación. Esta función es capaz de medir el desempeño de un modelo de aprendizaje automático entre 0 y 1, siendo el 0 la pérdida ideal de un modelo perfecto. Se definen dos variantes de la función de entropía-cruzada, por un lado la binaria, de la forma:

$$BCE = -(y \log(p) + (1 - y) \log(1 - p)) \tag{4.6}$$

donde p es la probabilidad dada para la predicción, e y es el indicador binario.

Por otro lado la categórica, escrita de la forma:

$$CCE = -\sum_{c=1}^{M} y_{o,c} \log(p_{o,c}) \tag{4.7}$$

siendo $y_{o,c}$=1 si la clase c es la predicción correcta para la entrada o; p_oc es la probabilidad que ha predicho el modelo para la clase c para la entrada o.

4.6.2. Descenso de gradiente

El objetivo de utilizar un algoritmo de descenso de gradiente es el de encontrar los pesos y los sesgos óptimos, de manera que minimicen el error calculado mediante la función de coste. Para lograrlo se desplaza iterativamente siguiendo la dirección de la pendiente más pronunciada, de acuerdo a lo definido por el negativo del gradiente. Al desplazarse repetidamente en esa dirección lleva a cabo el "descenso de gradiente", logrando en cada una de estas iteraciones disminuir el valor de la función de coste, llegando en algún momento a lo que de forma intuitiva denominaremos el mínimo.

La manera en que este método funciona puede dividirse en las siguientes etapas:

1. La primera etapa consiste en inicializar la función que se quiere optimizar. Para ello suele escogerse un punto aleatorio en el dominio, $\vec{x_0} = (x_1^0, ..., x_n^0)$.

2. El segundo paso es calcular el gradiente de la función de coste, expresada como f, a partir del punto calculado en la iteración previa, expresado como x_{t-1}:

$$\nabla(f(x_{t-1})) = (\frac{\partial f}{\partial x_1^{t-1}}, \frac{\partial f}{\partial x_2^{t-1}}, ..., \frac{\partial f}{\partial x_n^{t-1}}) \tag{4.8}$$

3. El tercer paso consiste en calcular el próximo punto en el que el valor de la función f sea menor. Para ello se hace uso de la siguiente ecuación:

$$x_t = x_{t-1} - \eta \nabla(f(x_{t-1})) \tag{4.9}$$

   en la cuál η representa la denominada tasa de aprendizaje, que refleja cuánto se avanza en cada iteración.

4. El último paso es repetir los pasos anteriores tantas veces como sea necesario, disminuyendo la función de coste hasta alcanzar un mínimo, o bien si se llega a un número máximo de iteraciones previamente estipulado.

En resumen, cuando aplicamos de forma iterativa el proceso anterior, lo que se consigue es ”bajar una pendiente” y encontrar un punto en el cual la función de coste genere un valor lo suficientemente bueno.

Cuando se aplica este método con el objetivo de minimizar el valor de la función de coste de una red neuronal, representada de la forma C(W,b), siendo W los pesos y b los sesgos, cada una de las iteraciones del proceso es denominada época o *epoch*. De esta manera, la actualización de los parámetros de la red puede expresarse de la siguiente forma:

$$w_{k_t+1} = w_{k_t} - \eta \frac{\partial C}{\partial w_{k_t}} \tag{4.10}$$

$$b_{k_t+1} = b_{k_t} - \eta \frac{\partial C}{\partial b_{k_t}} \tag{4.11}$$

4.6.3. Descenso estocástico de gradiente

Este método es una variante simplificada del algoritmo de descenso de gradiente. Destaca que en él, se escogen de forma aleatoria un pequeño conjunto de elementos para llevar a cabo la actualización de los pesos, en vez de usar todos los datos.

El método se estructura según los siguientes pasos:

1. Dividir de forma aleatoria el conjunto de datos en un numero M de subconjuntos de igual tamaño. Estos subconjuntos reciben la denominación de *mini-batches*.

2. Realizar una aproximación de la función de coste para cada *mini-batch*, en lugar de calcular el coste total, de la siguiente forma:

$$C(W,b) \approx \frac{1}{M} \sum_{j=1}^{M} C_j(W,b) \tag{4.12}$$

   en la cuál $C_j(W,b)$ representa el coste de un *mini-batch*.

3. Al utilizar la ecuación aproximada anterior, se calcula el gradiente y se actualizan los parámetros de la red de la siguiente manera:

$$w_{k_t+1} = w_{k_t} - \eta \frac{1}{M} \sum_{j=1}^{M} \frac{\partial C}{\partial w_{k_t}} \tag{4.13}$$

$$b_{k_t+1} = b_{k_t} - \eta \frac{1}{M} \sum_{j=1}^{M} \frac{\partial C}{\partial b_{k_t}} \tag{4.14}$$

4. Repetir los pasos anteriores tantas veces como sea necesario, disminuyendo la función de coste hasta alcanzar un mínimo, o bien si se llega a un número máximo de iteraciones previamente establecido.

La principal mejora que supone este método frente al descenso de gradiente estándar, es que el aprendizaje se ve acelerado debido a la mayor tasa de actualización de los parámetros. Al trabajar con subconjuntos aleatorios es mas complicado converger en un mínimo local, facilitando llegar al mínimo global.

4.6.4. Propagación hacia atrás

El algoritmo de propagación hacia atrás o *backpropagation* en inglés, es uno de los métodos más eficientes y potentes a la hora de realizar el cálculo de los gradientes de una función de coste, en relación a los parámetros que tiene la red. Aunque la idea detrás de este algoritmo existía desde los años 70, fue presentado formalmente en 1985 por Rumelhart et al.(53).

En el algoritmo de retropropagación se tiene una función de coste (p.ej. MSE o BCE) que se busca minimizar, de forma que el algoritmo de *backpropagation* se encarga de modificar y actualizar los pesos y los sesgos de la red en cada iteración o *epoch*.

Dentro del proceso de propagación de la información, se distinguen dos fases muy importantes, la fase *forward*, en la que las señales de información fluyen de forma natural desde las entradas a las salidas y la fase *backward*, en la que una vez obtenidas las salidas se procede a realizar la modificación de los pesos y sesgos, pero esta vez en sentido contrario.

El funcionamiento de este método puede expresarse en forma de tres pasos:

1. **Fase de inicio**: En el caso de partir sin datos iniciales sobre la red, los pesos y sesgos de la misma deben ser inicializados de forma aleatoria y seguidamente prepararse un patrón de entradas, generalmente seleccionado de forma aleatoria, correspondiente a una época.

2. **Fase de *forward***: En esta fase se lleva a cabo el proceso de propagación hacia delante. Partiendo de una parte de los datos correspondientes a la época n, se prepara como entrada a la red el vector $x(n)$. Se realiza entonces el proceso de *forward*, expresado en la ecuación:

$$v_j^{(L)}(n) = \sum_{i=0}^{m_0} w_{j,i}^{(L)}(n) y_i^{(L-1)}(n) \tag{4.15}$$

   Donde $v_j^{(L)}(n)$ representa el çampo local inducido", siendo j el número de neurona y L el número de la capa. La expresión $y_i^{(L-1)}$ simboliza el valor de salida de la neurona i, de la capa anterior a L durante la época n. Finalmente $w_{j,i}^{(L)}$ representa el peso existente entre la conexión de la neurona j de la capa L y la neurona i de la capa anterior.

   Una vez realizado este paso se podrá deducir el error, por medio de una función de coste. Usando una forma simplificada de lo que es la función de coste, la operación quedaría de la forma:

$$error_i(n) = e_i(n) - p_j(n) \tag{4.16}$$

   Siendo $e_i(n)$ el i-ésimo termino esperado para la salida $e(n)$, y $p_j(n)$ el resultado dado por la salida de la neurona j.

3. **Fase de *backward***: Tras calcularse los gradientes locales correspondientes mediante el uso del método de descenso de gradiente, los pesos deben ser ajustados mediante esta operación:

$$w_{j,i}^{(L)}(n+1) = w_{j,i}^{(L)}(n) + \alpha[w_{j,i}^{(L)}(n-1)] + \eta\delta_j^L(n) y_i^{(L-1)} \tag{4.17}$$

   Siendo η la tasa de aprendizaje o *learning rate*, $\delta_j^L(n)$ el gradiente local previamente calculado. El *learning rate* esta relacionado con la dirección que lleva el método de

backpropagation mediante el descenso de gradiente. El símbolo α hace referencia a la constante de *momentum*, que ayuda a reducir las oscilaciones que causa el hecho de que los pesos varíen.

Los pasos de *forward* y *backward* se repetirán tantas veces como sea necesario hasta alcanzar un mínimo, o bien si se llega a un número máximo de iteraciones previamente estipulado. Dado que la convergencia en un mínimo rara vez ocurre, es importante estipular cuando debe parar, p.ej. alcanzar un determinado valor de coste medio.

4.7. Medidas de prevención del sobreajuste

Cuando se entrena una red neuronal se busca que sea capaz de realizar la labor para la que fue entrenada, a partir de lo aprendido gracias al conjunto de datos de entrenamiento. Sin embargo, la red también debe tener la capacidad de enfrentarse a casos distintos a los que fue entrenada, es decir, deben poder generalizar.

Si una red neuronal se ajusta de forma excesiva a los datos con los que fue entrenada, aprenderá de forma muy detallada las características de éstos, perdiendo como consecuencia la capacidad de generalizar. Este fenómeno es el denominado *overfitting* o sobreajuste. Se trata de un problema bastante habitual, por lo que se han desarrollado muchas medidas que puedan evitar este comportamiento.

Una de las formas mas habituales para comprobarlo es dividir el conjunto de datos de entrenamiento en dos partes, una para entrenamiento y otra para validación. La red aprende con los datos de entrenamiento a lo largo de cada *epoch*, modificando los parámetros internos de sesgos y pesos, buscando minimizar el valor de la función de coste. El conjunto de validación no altera los parámetros de la red, pero sí permite comprobar el error.

Uno de los métodos mas utilizados para evitar el sobreajuste de un modelo de redes neuronales es el llamado método de *dropout*.

El método de *dropout* soluciona el problema del *overfitting* mediante la eliminación de neuronas de forma aleatoria dentro de una red profunda. Esto quiere decir, que el aporte de esas neuronas a la activación de las demás queda anulado temporalmente durante el paso de *forward* y que además estas neuronas no se vean afectadas por la actualización de parámetros durante el paso *backward*.

En el proceso de aprendizaje normal de una red neuronal profunda, el paso de backward genera relaciones de dependencia de una neurona con otra, beneficiando el resultado con los datos de entrenamiento, pero no con los datos distintos. El uso del método de *dropout* de forma aleatoria acaba con estas relaciones, haciendo que ciertas neuronas no sean tan importantes.

Se ha podido comprobar que el uso de este método ha mejorado el rendimiento de los modelos de redes neuronales en diversos campos de la inteligencia artificial, como por ejemplo, la visión artificial, la clasificación de textos o los análisis de voz.

La figura 4.8 muestra un esquema gráfico de aplicación del método de *dropout* de forma que en la parte superior se muestra un esquema general con todas las neuronas conectadas, mientras que en la parte inferior algunas de ellas han sido desconectadas.

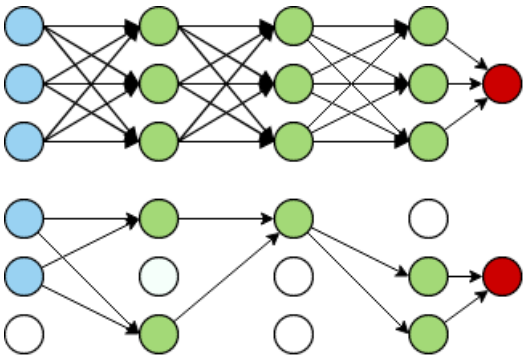

Figura 4.8: Red original (arriba) y red con *dropout* (abajo)

4.8. Redes neuronales convolucionales

Cuando se habla del término *deep learning* o aprendizaje profundo, el primer tipo de red que sale a relucir son las redes convolucionales. Este tipo de red neuronal, similar a las redes *feed-forward*, fundamentan su principal diferencia en la intervención de un tipo de capa denominada convolucional, que da nombre a la red. De esta forma puede decirse que las CNN (Convolutional Neural Networks) son un tipo de red neuronal que utiliza la operación de convolución, en vez de multiplicar matrices, en por lo menos una de las capas.

La figura 4.9 muestra un esquema general de un modelo CNN con las mencionadas capas, incluyendo además funciones de activación de tipo RELU, y capas de aplanamiento (Flatten) que pasan de volúmenes de datos a vectores o capas totalmente conectadas (Fully Connected) donde todas las neuronas están completamente conectadas entre capas.

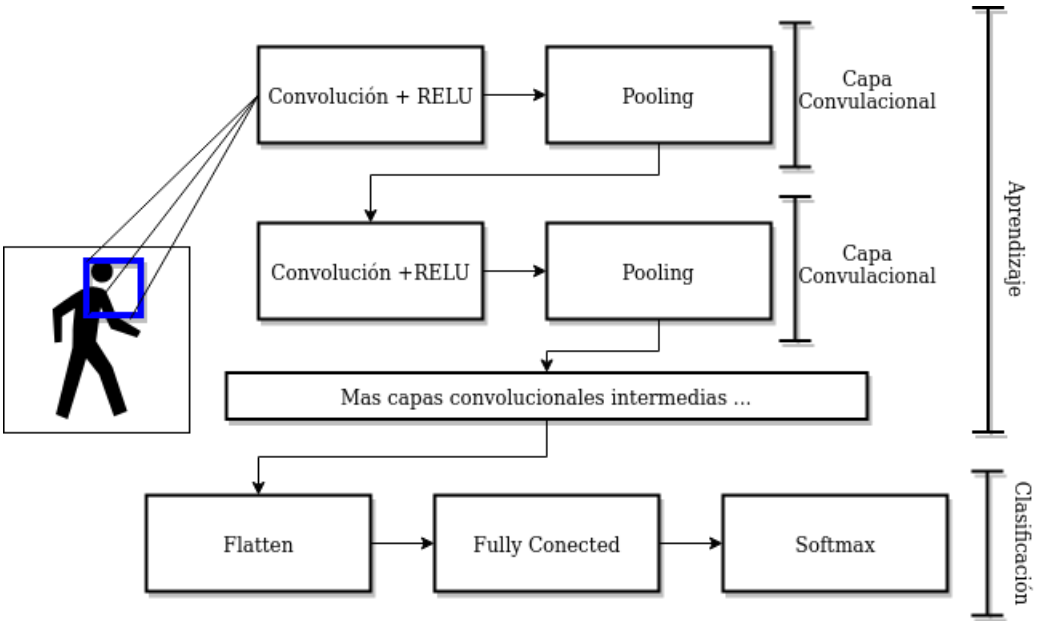

Figura 4.9: Arquitectura base de una red CNN

Este tipo de red divide y procesa los datos en partes más pequeñas, para más tarde combinar esta información en algunas de las capas mas profundas, permitiendo que las neuronas sean capaces de especializarse en ciertas regiones de los datos.

Han demostrado ser especialmente efectivas en problemas relacionados con el tratamiento de imágenes, la clasificación de textos o incluso el reconocimiento de voz.

Aunque la capa de convolución es la más representativa en una CNN, existen también otros tipos, destacando la llamada capa de *pooling* o agrupación, típicamente utilizada para realizar agrupaciones de píxeles sobre los resultados obtenidos tras la capa de convolución, y la capa de *softmax*, situada justo antes de la capa de salida donde se obtiene el resultado.

4.8.1. Capa de convolución

La operación de convolución parte de una serie de operaciones de sumas y productos entre la capa de entrada y el *kernel* (núcleo de convolución), permitiendo obtener un mapa de características de la imagen.

La operación de convolución puede expresarse de la forma:

$$s(t) = \int (x * w)(t) \tag{4.18}$$

El primer argumento, la x en la operación de convolución, es conocido como la entrada o *input*, y el segundo, la w, es el anteriormente nombrado *kernel*. Por otro lado s, es el mapa de características extraído de la imagen, también llamado *feature map* (mapa de características).

A la hora de utilizar esta operación durante una tarea de aprendizaje automático, la entrada es un vector o una matriz multidimensional de información y el *kernel*, otro vector o matriz multidimensional, compuesto de un conjunto de parámetros capaces de ajustarse durante la fase de aprendizaje. En el caso de una imagen I, la estructura del *input* es una estructura 2-D, que es procesada por un *kernel* K, también de dos dimensiones. Quedando de la forma:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n) \tag{4.19}$$

-

Estas capas aplican en paralelo varias operaciones de convolución en distintos píxeles (i,j). Para comprender cómo se realiza este paso, lo mejor es pensar en la aplicación del *kernel* de la misma forma que lo haría una ventana deslizante, desplazándose por toda la imagen y calculando el mapa de características correspondiente en cada movimiento, figura 4.10.

Aunque está bastante generalizado trabajar con convoluciones 2-D, también existen de tipo 3-D, de manera que el *kernel* tendrá en este caso la forma de un cuboide, este es un caso normal cuando se trabaja con imágenes RGB[ref], en las cuales existe una tercera dimensión apilando los tres canales espectrales en el modelo de color RGB. Esto puede generalizarse a un número de dimensiones N, como es por ejemplo el caso de algunas de las redes convolucionales utilizadas en reconocimiento de vídeo, en las que existe una dimensión más para el tiempo.

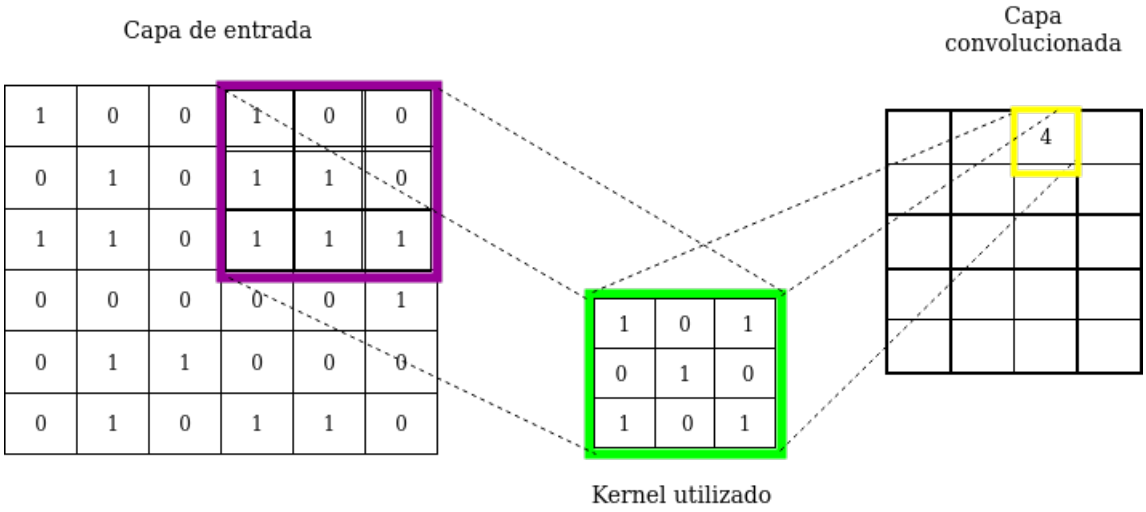

Figura 4.10: Ejemplo de un paso de convolución 2D sobre una imagen binaria utilizando un kernel 3x3

4.8.2. Capa de pooling

Uno de las capas mas utilizadas dentro del ámbito de las CNN, aparte de la capa convolucional, es la capa de *pooling* o agrupamiento. Generalmente esta capa se utiliza para reducir el tamaño de representación de los datos, permitiendo así aumentar la velocidad de cómputo.

Para llevar a cabo una operación de *pooling*, primero debe definirse el tamaño del filtro que se aplicará. Posteriormente la imagen de entrada o matriz de datos, se divide en secciones del mismo tamaño que el filtro. Sobre cada una de estas secciones se lleva a cabo la operación de *pooling*, generalmente existen dos variantes: *max-pooling* y *average-pooling*. La variante de *max-pooling* comprueba los valores de los píxeles en cada sección y elige el máximo, en cambio, *average-pooling* devuelve la media numérica de los valores en la ventana o filtro, figura 4.11.

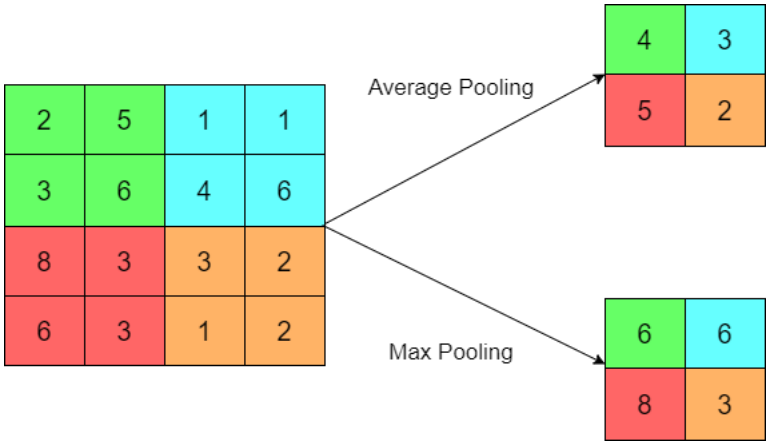

Figura 4.11: Ejemplo de *pooling* utilizando un filtro 2x2

4.8.3. Capa de softmax

La capa *softmax* se introduce con el objetivo de aplicar sobre los datos la denominada función exponencial normalizada, capaz de "traducir" un vector M-dimensional, x, conformado por valores reales arbitrarios, en un vector M-dimensional, S(x), formado por valores dentro del rango [0, 1], siendo S la función *softmax*.

$$S(j) = \frac{e^{z_j}}{\sum_{i=1}^{M} e^{z_i}} \tag{4.20}$$

$$z_j = w_j^T * x \tag{4.21}$$

La figura 5.12, muestra un esquema sintético de la capa softmax, observándose a la salida el tipo de clase (camión, coche, furgoneta o moto, entre otras) asignado a un vector de entrada, además de la probabilidad de pertenencia a la clase correspondiente

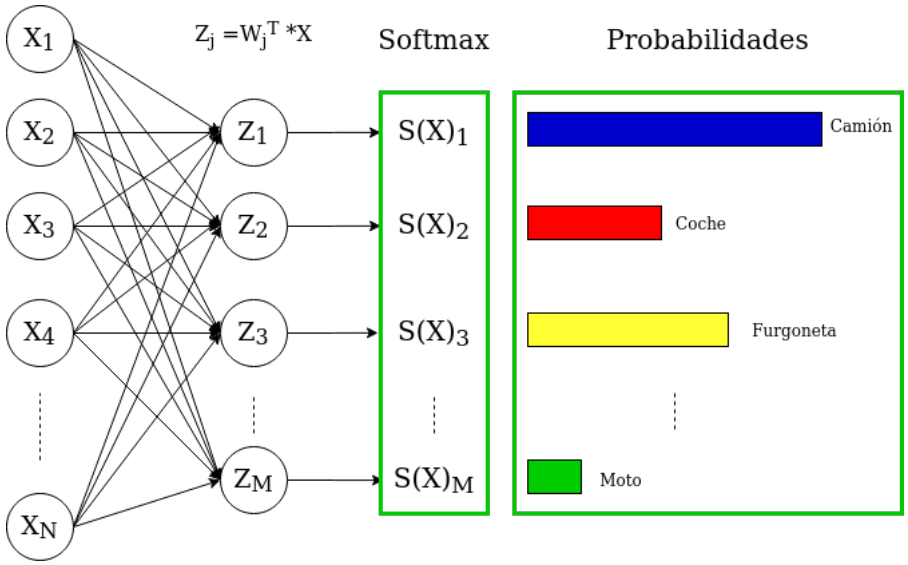

Figura 4.12: Ejemplo de capa *softmax*

Este tipo de capa permite de obtener las probabilidades asignadas a cada una de las diferentes clases entre las que se puede clasificar un elemento. Resulta de especial importancia recalcar que la capa *softmax* tiene tantos nodos de salida como clases posibles existen.

Capítulo 5

Gestión del trabajo

5.1. Introducción

A la hora de realizar un proyecto pueden surgir un gran número de contratiempos, que es importante ser capaz de gestionar. Por ello, es necesario llevar una planificación detallada que permita administrar cada una de las tareas en las que un proyecto se divide. Concretamente, existen tres puntos objetivos en toda planificación de proyecto:

- Capacidad de gestionar los tiempos de comienzo y evolución de las partes del proyecto.
- Capacidad de administrar y contener los posibles problemas que puedan surgir.
- Tener un enfoque destinado a facilitar la finalización del proyecto con los mejores resultados posibles.

Generalmente para poder realizar el proyecto ajustándose a los objetivos anteriores, resulta de especial importancia la utilización de herramientas externas. De esta forma, para este proyecto se ha considerado prioritario la inclusión de herramientas que permitan cumplir una gestión adecuada del mismo. En especial, las herramientas utilizadas con tal propósito, cubriendo los diferentes aspectos del trabajo, han sido las siguientes:

- Gestión de tiempos y tareas: Trello.
- Control de versiones del código: Github.
- Almacenamiento y copias de seguridad: Google Drive.

A continuación se detallan las herramientas anteriores, no aquellas herramientas y tecnologías relacionadas con el desarrollo del código del proyecto en sí, que se encuentran detalladas en la sección 3.4.

5.2. Herramientas de gestión del trabajo

5.2.1. Trello

Trello(11) es una herramienta software de gestión de proyectos en línea, realmente útil a la hora gestionar diferentes tareas y trabajos. Permite realizar seguimientos detallados de los avances de un trabajo, generar alertas de fechas límites y una gran variedad de otras funcionalidades que facilitan la organización eficiente de un proyecto.

La idea detrás del funcionamiento de *Trello* es la organización del trabajo siguiendo un estilo de tableros, columnas y tarjetas, emulando en un entorno virtual la organización tradicional de proyectos en tablones de corcho y chinchetas.

En *Trello*, los tablero se dividen en columnas, que son una serie de listas ordenadas utilizadas para dividir la información en función de diferentes criterios de clasificación personalizables. Estas columnas, están formadas por las "tarjetas", que representan cada una de las tareas individuales que conforman el proyecto. Pueden ser etiquetadas para diferenciar a qué campo pertenece cada una.

En el ámbito de este proyecto la división en tableros ha sido la siguiente:

- **Detección de personas**: administración de cada una de las tareas que iban surgiendo relacionadas con la parte de detección de personas en vídeo.
- **Localizador de personas**: tareas destinadas a investigar e implementar el sistema de *tracking* de personas.
- **Reconocimiento de acciones**: gestión del trabajo relacionada con la investigación e implementación de la parte del sistema dedicada al reconocimiento de las acciones humanas.
- **Integración y pruebas finales**: administración de la unificación final de las partes anteriores en un único sistema y de la realización de pruebas de funcionamiento.

Diferenciando las "tarjetas"de cada tablero por medio de etiquetas, en este caso referidas a los tres tipos de actividad presente en cada módulo del proyecto: investigación, planificación, implementación y pruebas. La figura 5.1 muestra el tablero del módulo detección de personas en Trello.

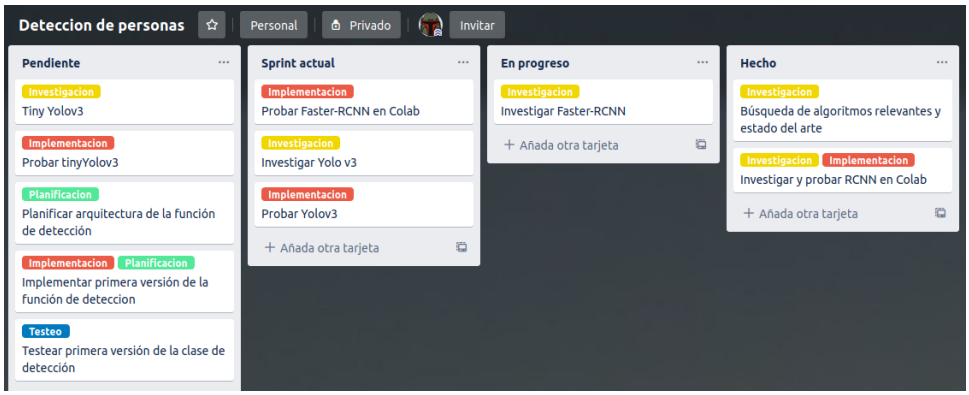

Figura 5.1: Muestra del tablero en *Trello* del módulo de detección de personas

5.2.2. Github

Es una reconocida plataforma para alojar proyectos y gestionar sus diferentes versiones mediante Git (40). Fue lanzada en 2008 (1) de la mano de *Ruby on Rails*(10). Debido a la rápida popularidad y gran repertorio de funcionalidades de la herramienta, fue comprada por *Microsoft* en 2018 por varios miles de millones de dólares.

Permite trabajar de forma colaborativa entre programadores, facilitando las actualizaciones de código. Las principales razones que nos han llevado a usarlo han sido entre otras: su popularidad actual, siendo de hecho una seña de identidad en muchos currículum, las funcionalidades que ofrece, y especialmente que es una de las herramientas de control de versiones mas utilizadas a nivel académico.

Su utilización resulta indispensable para realización de cualquier proyecto de software. No obstante, al tratarse de un proyecto individual, no se han visto aprovechadas la gran variedad de funcionalidades que *Github* puede ofrecer, estando la gran mayoría de ellas enfocadas al trabajo colaborativo de equipos de desarrollo.

5.2.3. Google Drive

Es un servicio de almacenamiento en la nube creado por *Google* y lanzado en 2012 (8). Permite crear, gestionar y compartir archivos desde una interfaz web, sin tener que descargarlos en local. Permite acceder en todo momento a la información almacenada desde cualquier tipo de dispositivo con acceso a internet, lo que posibilita hacer uso de los datos desde cualquier ubicación.

Aparte de todas las posibilidades que brinda su almacenamiento, dispone de una serie de aplicaciones muy similares a las del paquete *Microsoft Office*, lo que permite leer y editar documentos siempre que se necesite.

Google Drive es gratuito, siempre y cuando se disponga de una dirección de correo de *Gmail*, no obstante, esta versión tiene un limite de almacenamiento de 15 GB. En caso de necesitar más cantidad, dada la facilidad de escalada de la herramienta, es tan fácil como aumentar el plan de almacenamiento a cambio de un reducido coste.

En el caso particular de la Universidad Complutense de Madrid, las cuentas de correo universitarias permiten un almacenamiento sin límites, siendo de gran ayuda a la hora de almacenar grandes conjuntos de datos.

Capítulo 6

Sistema de videovigilancia inteligente

6.1. Introducción

Tal y como se explica en los capítulos previos, el objetivo final de este proyecto es el diseño e implementación de un sistema que, mediante la aplicación de diferentes técnicas del campo de la visión por computador y del aprendizaje profundo, permita realizar labores de videovigilancia inteligente, siendo capaz de detectar humanos, localizarlos y extraer y clasificar las acciones que estos están realizando.

La labor de un sistema de estas características, no solo facilita las labores de supervisión de cámaras de seguridad, tradicionalmente realizadas por el ojo humano, sino que es capaz de detectar y clasificar el comportamiento, permitiendo realizar seguimientos de tendencias entre la población.

En el caso de una situación como la relativa a la pandemia Covid-19, detectar y localizar focos de aglomeraciones de personas sin respetar las distancias de seguridad necesarias, permite alertar a las autoridades competentes para tomar una decisión.

Si bien un mayor refinamiento de la faceta de reconocimiento de acciones, podría ser capaz de evitar ciertos comportamientos de naturaleza criminal, al poder detectarlos y tomar contramedidas lo antes posibles. Incluso con la tecnología y diseño adecuado, podrían predecirse cierto tipo de actividades que pudieran derivar en sucesos criminales, facilitando la labor de las fuerzas del orden, aumentando la seguridad de los ciudadanos.

Por tanto, según lo anterior, no cabe duda que la implantación de este tipo de sistemas inteligentes en diversos y variados escenarios facilita tareas de videovigilancia según su finalidad. En las secciones que conforman este capítulo se expone el diseño del funcionamiento general del sistema y de los tres módulos que lo conforman.

6.1.1. Funcionamiento general del sistema

En el sistema desarrollado se diferencian tres partes, cada una de ellas relacionada con diferentes vertientes de la visión artificial y el aprendizaje automático. Estas partes, identificadas como módulos son las siguientes, figura 6.1:

1. **Módulo de detección de humanos**: Detecta a los humanos presentes en un vídeo, en base a una serie de umbrales de precisión, destacándolos mediante cajas delimitadoras.

2. **Módulo de rastreo y localización de humanos**: Asigna identificadores únicos a cada humano, y se cerciora de localizar la posición del humano en el vídeo, manteniendo un seguimiento del mismo. Gestiona todas las labores relacionadas con la detección de nuevos humanos, predicción de sus movimientos y desaparición, así como de la segmentación de sus movimientos para posteriormente usarlos para reconocer sus acciones.

3. **Módulo de reconocimiento de acciones**: Reconoce la acción realizada por un humano en base a la información recibida del modulo anterior.

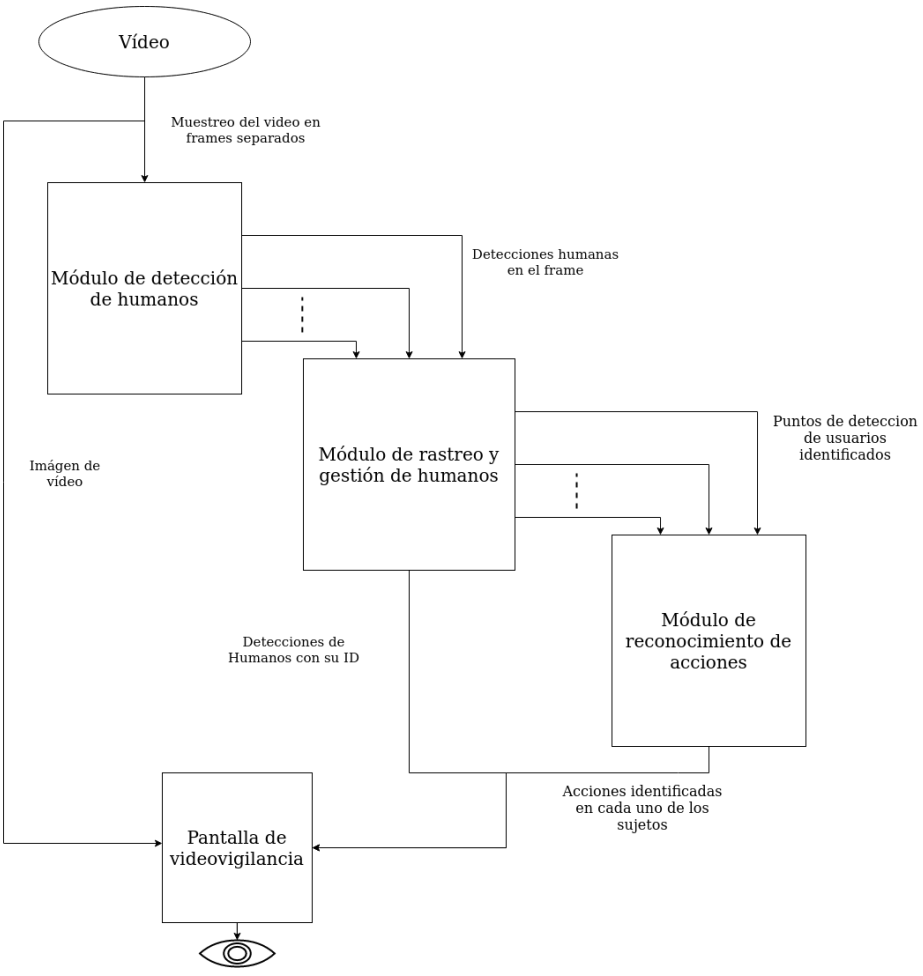

Figura 6.1: Diagrama que representa el flujo de funcionamiento general del sistema

Merece la pena recalcar, que tal y como está implementado el sistema, la mayoría de funcionalidades de un módulo dependen a su vez del módulo anterior, generando un flujo de dependencias en cascada, tal y como aparece representado en la figura 6.1, en la cual también está representada la estructura de interconexión entre módulos del sistema.

6.2. Módulo de detección de humanos

Como parte inicial a la hora de poder realizar labores de videovigilancia inteligente, resulta de especial prioridad desarrollar la capacidad de detectar de forma automática la presencia de personas en un determinado espacio.

El sistema necesita de un vídeo de entrada, sobre el cual se van a llevar a cabo una serie de procesos que darán como resultados las personas detectadas en el mismo.

Llevando a cabo una división de los procesos presentes en este módulo, se pueden encontrar las siguientes etapas y sub-etapas:

1. **Muestreo del vídeo y preprocesado de *frames***
   - *a*) Extracción de *frame* del vídeo
   - *b*) Procesado de imagen y transformación en *blob*
2. **Generación de detecciones y clasificación**
   - *a*) Inferencia sobre el modelo Yolov3
   - *b*) Procesamiento de la salida de la red
   - *c*) Pre-filtrado de las detecciones encontradas
3. **Filtrado por niveles de confianza**
   - *a*) Almacenamiento de las detecciones
   - *b*) Aplicación del filtrado por NMS (Non-Maximum Supresion)

En primer lugar el sistema accede al vídeo de entrada, sobre el cual se quiere realizar la videovigilancia inteligente. Este vídeo será muestreado *frame* a *frame* en forma de bucle, asegurando la integridad de cada uno de los *frames* extraídos de forma individual, eliminando aquellos *frames* corruptos que pudieran causar problemas en etapas mas avanzadas del sistema.

Para ello cada vez que un *frame* es extraído se evalúa una variable de control de lectura, facilitada por las herramientas de captura de vídeo presentes en la librería OpenCV. Posteriormente a esta comprobación, las dimensiones de *frame* son comparadas con las dimensiones esperadas de la entrada de vídeo, en caso de no coincidir el sistema, no realizará ningún tipo de acción sobre este *frame* y procederá a extraer el siguiente.

Una vez se ha extraído un *frame* cumpliendo los requisitos anteriores, éste es preprocesado y transformado en formato *blob*. Un *blob* es potencialmente una colección de imágenes de igual tamaño, mismas dimensiones espaciales y mismo número de canales de color, que deben ser procesadas de la misma forma.

En este caso, el preprocesado llevado a cabo sobre la imagen, se hace con el objetivo de generar una entrada válida para el modelo de red neuronal utilizado. En concreto sobre la imagen se han aplicado las siguientes operaciones de preprocesado:

- Redimensión de la imagen a 416 x 416, siendo éste el tamaño esperado como entrada por red neuronal
- Intercambio de los canales de color R y B, pasando de BGR a RGB.
- Escalado del espacio de la imagen en un rango particular, concretamente $\sigma = 255.0$

Una vez la imagen ha sido preprocesada, es proporcionada como entrada al modelo de detección de objetos.

De entre los métodos de detección de objetos citados en la seccion 3.3, se consideran aspirantes a formar parte del sistema de videovigilancia, únicamente a:

- Faster-RCNN, descrito en la sección 3.3.3 .
- Yolo versión 3, descrito en la sección 3.3.4 .

El algoritmo Faster-RCNN, pese a situarse como uno de los mejores algoritmos existentes en el estado del arte actual, comete mas errores de fondo de imagen a la hora de detectar los de objetos, en cambio Yolov3 no. Además, pese a alcanzar grandes precisiones, Faster-RCNN tiene una velocidad de procesamiento que hace prácticamente inviable su uso en una labor que pretenda trabajar a velocidades cercanas al procesamiento en tiempo real. Por lo contrario, Yolov3 demuestra ser capaz de alcanzar grandes precisiones manteniendo unas muy altas tasas de *frames* por segundo procesados.

El modelo finalmente utilizado tras investigar y probar a pequeña escala ambos métodos, ha sido el algoritmo de detección de objetos Yolo, en su versión 3. Esta versión, a diferencia de versiones anteriores de Yolo, utiliza una arquitectura de modelo de red neuronal DarkNet-53, figura 6.2, más profunda y compleja que sus predecesoras, pero capaz de mejorar las detecciones de objetos de menor tamaño y de generalizar mejor.

	Tipo	Filtros	Dimensión/ *stride*	Salida
	Convolucional	32	3×3	256×256
	Convolucional	64	3×3/2	128×128
1×	Convolucional	32	1×1	
	Convolucional	64	3×3	
	Residual			128×128
	Convolucional	128	3×3/2	64×64
2×	Convolucional	64	1×1	
	Convolucional	128	3×3	
	Residual			64×64
	Convolucional	256	3×3/2	32×32
8×	Convolucional	128	1×1	
	Convolucional	256	3×3	
	Residual			32×32
	Convolucional	512	3×3/2	16×16
8×	Convolucional	256	1×1	
	Convolucional	512	3×3	
	Residual			16×16
	Convolucional	1024	3×3/2	8×8
4×	Convolucional	512	1×1	
	Convolucional	1024	3×3	
	Residual			8×8
	Average pool	Global		
	Conectada	1000		
	Softmax			

Figura 6.2: Arquitectura de red DarkNet-53

Una vez ejecutado un paso *forward* sobre Yolov3, devolverá una salida en forma de lista, cuyos elementos presentan la siguiente forma:

$$(coordenada_x, coordenada_y, anchura, altura, probabilidades[..]) \tag{6.1}$$

Donde los dos primeros parámetros hacen referencia a las coordenadas (x, y) de la caja delimitadora de un objeto, y los dos segundos a sus dimensiones. El último parámetro es una lista de probabilidades de tamaño N, siendo N el número de objetos diferentes que el modelo es capaz de diferenciar, en este caso 80, al estar pre-entrenado con el *dataset* COCO(39).

Cada uno de los elementos de la lista hace referencia a la probabilidad de que dicha detección sea de ese tipo de objeto, siendo el objeto con mayor probabilidad el que más posibilidades tiene de ser el que está presente en la detección.

Una vez estas predicciones han sido obtenidas, son procesadas siguiendo los siguientes pasos:

1. Extracción del índice de mayor probabilidad de la lista de probabilidades.
2. Almacenamiento de la probabilidad extraída anteriormente.
3. Pre-filtrado de las predicciones con probabilidades más débiles que no superen el umbral determinado, omitiendo así los siguientes pasos. En este caso se aplica un umbral de 0.6.
4. Traslación de las coordenadas y dimensiones a coordenadas relativas a la imagen del *frame*, permitiendo así generar cajas delimitadoras.

Una vez las detecciones han sido procesadas y transformadas a un formato de cajas delimitadoras, y teniendo su probabilidad más alta almacenada, el sistema procede aplicar una función de supresión no máxima, o NMS (Non Max Suppression), ya que Yolov3 no la aplica.

Al aplicar la función NMS, se está llevando a cabo no solo un filtrado por niveles de confianza de las predicciones que pasaron los pasos del post-procesamiento, sino que además suprime los cuadros delimitadores que se superponen significativamente, manteniendo únicamente los más precisos.

Finalmente los cajas delimitadoras definitivas son filtradas, aceptando únicamente aquellas clasificadas como Personas. Estas cajas delimitadoras son enviadas al módulo dos, que hará uso de ellas para rastrear y mantener una gestión de las personas bajo seguimiento, figura 6.3.

Figura 6.3: Ejemplo de aplicación de NMS a una imagen (6)

La figura 6.4 muestra la estructura general de funcionamiento del módulo de detección.

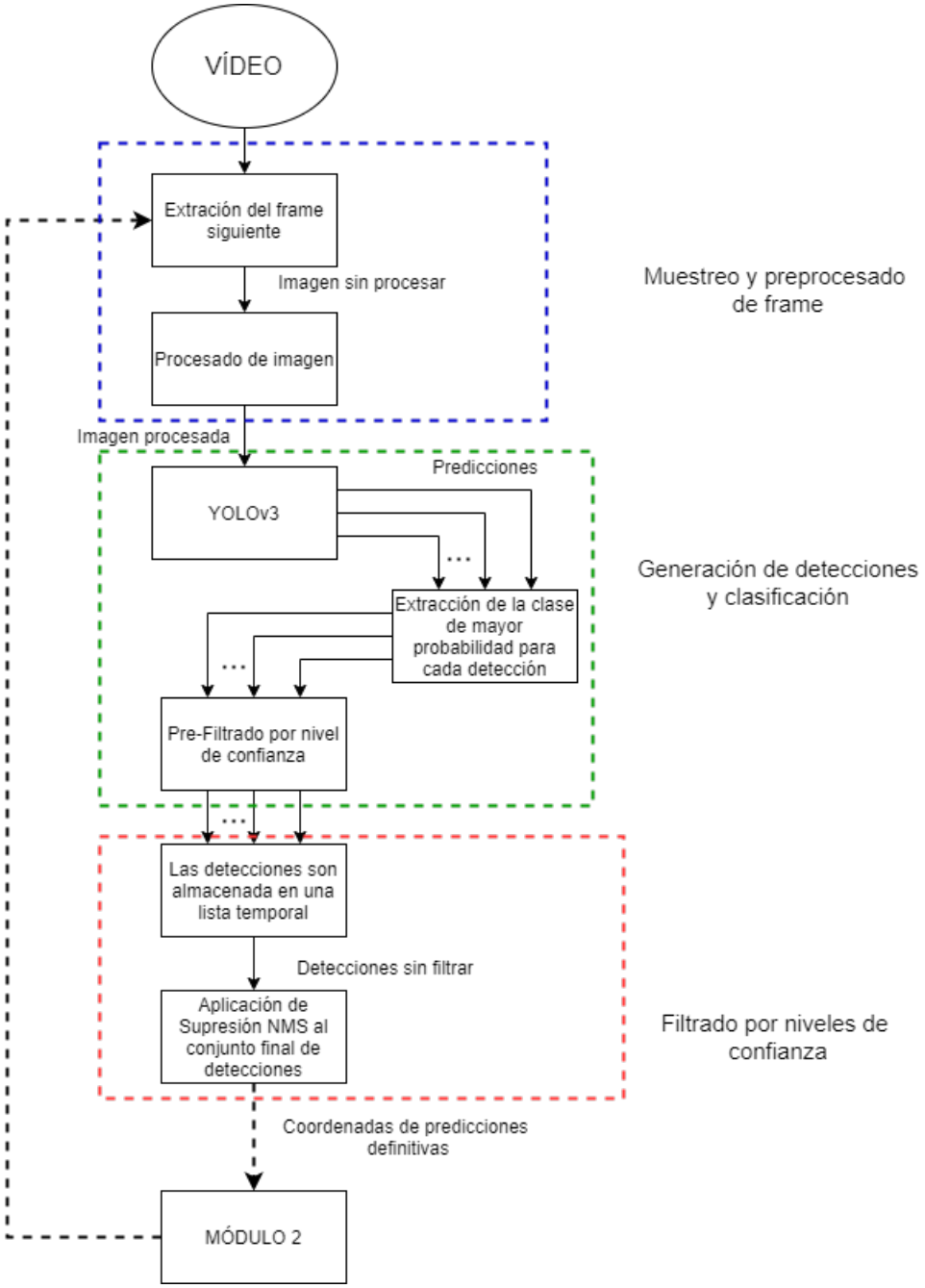

Figura 6.4: Estructura de funcionamiento del módulo de detección

6.3. Módulo de localización y gestión de humanos

A la hora de crear un sistema capaz de realizar labores de videovigilancia inteligente, no solo es necesario que éste sea capaz de detectar personas en cada uno de los *frames* en los que se divide un vídeo, sino que debe ser capaz de relacionar las detecciones de un *frame* con las del siguiente. Al relacionar las personas detectadas en un *frame* con las detectadas en el *frame* siguiente, el sistema puede realizar un identificación individual de las personas que hay en la escena, siendo capaz de mantener un seguimiento individual de cada una de las personas y asignarlas un identificador único.

Cuando se realiza seguimiento sobre una persona identificada, pueden realizarse mapas de trazas de sus últimas posiciones y predecir cuál será su siguiente localización, así como realizar extracciones de sus movimientos individuales, dentro de un espacio temporal consecutivo.

El proceso de seguimiento y gestión de usuarios presente en este módulo, se divide en varios pasos:

1. **Procesamiento de la entrada**
   - Gestión de la ausencia de nuevas detecciones
   - Gestión de la ausencia de humanos bajo seguimiento
   - Extracción de los centroides a partir de los datos de entrada.
2. **Cálculo de correspondencias**
   - Cálculo de distancias euclídeas de forma matricial
   - Obtención de parejas de distancias mínimas aspirantes a crear una correspondencia.
   - Filtrado por distancia máxima
3. **Gestión de las correspondencias**
   - Actualización de localizaciones de humanos bajo seguimiento
   - Almacenamiento de últimas localizaciones
   - Envio de información al módulo tres
   - Predicción y guardado de futuras posiciones
4. **Gestión de las no correspondencias**
   - Añadir nuevos humanos a seguimiento
   - Realizar seguimiento sobre las desapariciones
   - Eliminar del sistema los seguimientos que llevan desaparecidos demasiados frames consecutivos.

Como idea principal a la hora de entender el funcionamiento de este módulo, hay que decir que él mismo gestiona una lista de "personas", es decir, aquellas personas que tiene bajo seguimiento. De esta manera, puede asignar a cada una de ellas un identificador único y comprobar cuándo siguen estando en la imagen o cuándo han desaparecido del campo de visión de la cámara.

La mayoría de los procesos descritos a continuación, persiguen como objetivo gestionar de manera óptima esta lista de "personas", actualizándola y almacenando en ella los datos necesarios en cada situación. La arquitectura de funcionamiento de este módulo se describe en la figura 6.7.

En primer lugar se reciben las detecciones del nuevo frame en forma de cajas delimitadoras, tal y como se explica en el apartado 6.2. Estas cajas pertenecen a un *frame* del vídeo en particular, el cual también es recibido como una de las entradas del módulo de seguimiento.

Lo primero que hace el sistema es comprobar que la lista de cajas no está vacía. En el caso de que la lista de cajas sí se encuentre vacía, significará que durante ese *frame* no hay personas en la imagen. Como consecuencia, el sistema buscará a todas y cada una de las personas que ya tenga bajo seguimiento en la lista y actualizará su contador de desapariciones, incrementándolo en 1. En el momento que este contador llegue a un límite previamente definido, se llevarán a cabo una serie de acciones descritas más adelante. En el caso de que no existan personas bajo seguimiento, no se realizarán más acciones y el flujo de ejecución de este módulo finalizará.

Si por el contrario, la lista de cajas no está vacía, se comprueba si la de seguimiento lo está. Si la de seguimiento está vacía todas las cajas serán entendidas como nuevas detecciones de humanos, y serán incluidas en la lista de seguimiento, tras haber sido transformadas en objetos de tipo Persona. Un objeto de tipo Persona consta de la siguiente información:

- **ID**: dado por el sistema de forma iterativa y única.
- **KF**: inicialización del algoritmo de Filtro de Kalman de forma independiente para predecir las posiciones de cada Persona.
- **Centroide**: posición (x,y) actual de la Persona.
- **Predicción**: posición predicha para la Persona, en un inicio es la misma que la actual.
- **Desapariciones**: iniciada a 0, es el contador de *frames* consecutivos que una Persona lleva desaparecida.
- **Traza**: lista finita de últimas posiciones de la Persona.

Para ello, de las coordenadas de las cajas se extraen sus centroides y se procede a inicializar las nuevas Personas, siguiendo el formato expresado en la lista anterior.

Dado que para poder realizar predicciones de los siguientes movimientos necesitaremos al menos tener más de una posición, el flujo de ejecución en este módulo finalizará aquí.

Si existen nuevas detecciones y además hay Personas bajo seguimiento, deberá realizarse un proceso de búsqueda de correspondencias.

El proceso de búsqueda de correspondencias tiene como objetivo generar una serie de relaciones entre los centroides de las cajas de la entrada y los centroides de las personas bajo seguimiento. La creación de estas relaciones dará pie a encontrar las posiciones a las que las Personas bajo seguimiento se han desplazado en el trascurso de un *frame*.

La forma en la que el sistema calcula los aspirantes a correspondencias consiste en calcular las distancias euclídeas. Se calculan estas distancias entre los centroides de las nuevas detecciones y las coordenadas predichas para cada persona bajo seguimiento.

Con el uso de la distancia euclídea se calculan las distancias entre los pares de puntos reales en la imagen, tal y como lo requiere el sistema. El cálculo de la distancia euclídea se define de la siguiente forma:

$$d(x, y) = \sqrt{\sum_{i=1}^{N}(x_i - y_i)^2} \tag{6.2}$$

El resultado de calcular las distancias euclídeas entre un vector de nuevas detecciones de tamaño N y el vector de personas bajo seguimiento de tamaño M, devuelve una matriz NxM con las distancias entre todos los puntos.

Una vez esta matriz ha sido calculada, para cada uno de las personas bajo seguimiento se selecciona de entre las nuevas detecciones las que disten una menor distancia. Este proceso se realiza de forma similar, y en gran medida inspirado en el funcionamiento del algoritmo húngaro(36).

En la figura 6.5 se representa un esquema de visualización de las coordenadas de nuevas detecciones en un*frame* N+1, frente a las predicciones de posición de las personas bajo seguimiento del *frame* N. Como se observa, las posiciones predichas para las personas en el *frame* N+1 son bastante precisas frente a las posiciones reales.

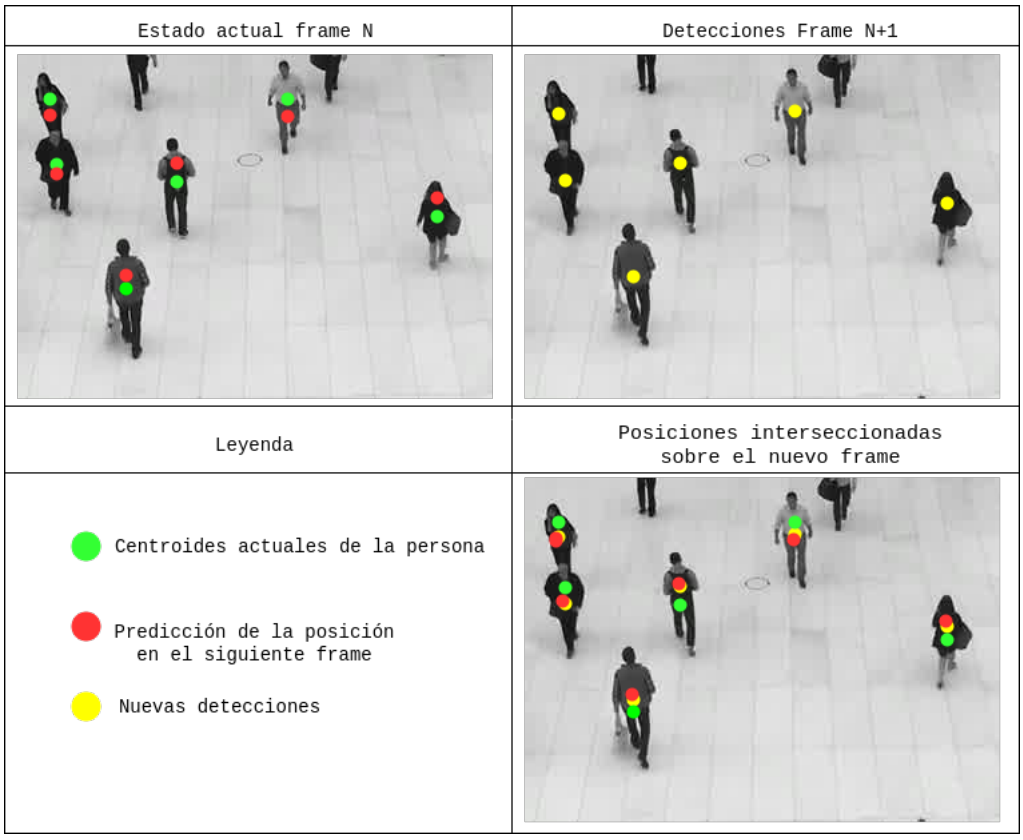

Figura 6.5: "Interseccion" entre las nuevas detecciones y las predicciones de las personas bajo seguimiento, sobre un vídeo de la plataforma Pexels(6).

Una vez aplicado el método del algoritmo húngaro, usando las distancias euclídeas como medida de distancia, también denominado coste, entre los puntos detectados y los

puntos esperados de la imagen, tendríamos la confirmación de la existencia de relaciones de correspondencia entre dichos puntos.

A raíz de esto, se generan una serie de parejas $(Persona, NuevaDeteccin)$ entre puntos con las menores distancias unos de otros. Estas parejas de puntos se presentan como aspirantes a generar una relación de correspondencia y son examinadas. Aquellas parejas cuyos elementos disten una distancia mayor a la máxima aceptada no serán procesadas como correspondencias.

Si bien, aquellas parejas que cumplan el requisito de distancia máxima entre ambos puntos, pueden no ser consideradas correspondencias si cualquiera de los dos elementos que forman la pareja ya ha sido asignado anteriormente a una pareja con menor distancia de separación.

Para mantener un control de asignación de correspondencias que evite duplicados, las nuevas correspondencias marcan a ambos miembros de la pareja como utilizados, evitando que sean usados más adelante.

A continuación, aquellas parejas que han pasado el filtro anterior, son sometidas a una serie de procesos que forman la denominada etapa de gestión de correspondencias.

El primer paso para gestionar una correspondencia exitosa, es la actualización. Al considerar que una pareja $(Persona, NuevaDeteccin)$ tiene una relación de correspondencia, lo que significa es que ambas hacen referencia a la misma persona en un diferente espacio temporal. El sistema implementado sustituirá los valores de posición de la persona bajo seguimiento, es decir su centroide, por los de la nueva detección.

Tras actualizar la posición de la persona, el siguiente paso será predecir la posición futura en la que con gran probabilidad la persona podría situarse en la imagen.

Para llevar a cabo el proceso de predicción, como se ha mencionado anteriormente, se ha utilizado un algoritmo denominado Filtro de Kalman(32). Este algoritmo es utilizado para predicción de valores en una gran diversidad de campos, incluyendo los financieros y para seguimiento de aviones por ejemplo. No obstante, ha alcanzado una gran popularidad en el campo de la predicción de movimientos en vídeos.

En el caso del sistema, este algoritmo es utilizado para predecir las siguientes posiciones de una persona. Para calcularlas se parte de las posiciones anteriores en el tiempo, para calcular la dirección de movimiento, aceleración y velocidad de desplazamiento. Este algoritmo es adaptable a cambios de velocidad y dirección dada su capacidad de corrección, mediante la cual el algoritmo se alimenta de nuevas posiciones en cada intervalo temporal (*frame*) y las utiliza para corregir las predicciones.

Tal y como se explica anteriormente, la ejecución de la predicción sobre la nueva posición de la persona bajo seguimiento, ejecutará primero un paso de corrección con esta nueva posición y posteriormente uno de predicción. El resultado de ese paso de predicción serán las futuras coordenadas esperadas para el siguiente *frame*, que serán almacenadas en el campo de predicción de la persona.

Tras el paso anterior se realiza la actualización de la lista de trazas, también denominado guardado de últimas localizaciones, es decir, la nueva posición de la persona será añadida a una lista de un tamaño predefinido. El hecho de mantener un almacenamiento de las últimas posiciones, brinda la oportunidad de presentar en la pantalla la trayectoria de movimiento que ha seguido una persona.

En la figura 6.6 se observan las trazas dejadas por dos personas montando en bicicleta por una calle bajo el control del sistema de videovigilancia inteligente. En esta versión los puntos de las trayectorias no están normalizados, presentando un aspecto de bordes irregulares al depender de la precisión de la detección.

Figura 6.6: Ejemplo de funcionamiento del pintado de trayectorias en un vídeo(6).

Una vez la lista de trazas ha sido actualizada, este módulo del sistema procede a actualizar los frames de movimiento de cada persona bajo seguimiento. Para ello hace uso del frame de vídeo que previamente recibió como parte de las entradas y de las nuevas coordenadas de la persona. Esta información es enviada al módulo de predicción de acciones, explicado en el apartado 6.4, el cuál se encargará de procesarla y extraer de ella las partes necesarias para poder gestionar la predicción de acciones humanas.

Finalmente tras terminar el procesamiento de las correspondencias confirmadas, deben ser procesadas aquellas detecciones y personas bajo seguimiento que no lo fueron previamente, al no formar parte de ninguna de las relaciones de correspondencia existentes.

De entre ellas, podemos distinguir dos tipos:

- Detecciones que no corresponden con ninguna persona existente.
- Personas bajo seguimiento que no están en el nuevo *frame*.

En el primer caso que ocurra algo así quiere decir que la persona a la que hace referencia la detección no esta bajo seguimiento, es decir, es nueva en el sistema. De manera que el procedimiento seguido por el sistema es el de inicializar y añadir una nueva persona a la lista de personas bajo seguimiento.

En el segundo caso, esto puede ocurrir por dos razones: la primera que la persona haya salido del campo de visión de la cámara y no pueda ser detectada por el módulo de detección, y la segunda, que debido a un error de detección del módulo de detección no se haya detectado una persona en ese punto. Generalmente ese último error puede ocurrir por superposiciones de personas que se encuentran muy cercanas unas de otras.

Cuando esto ocurre el sistema accede al contador de desapariciones de la persona en cuestión y lo incrementa. Posteriormente comprueba si el incremento ha alcanzado el máximo permitido de *frames* desaparecidos consecutivos, de ser así la persona será eliminada tanto de las listas del módulo de rastreo como del módulo de predicción de acciones.

Para solventar pérdidas de detección puntuales por superposiciones de personas, se pueden tomar dos opciones, que implican asumir una u otra de las siguientes declaraciones:

- Asumir que cuando desaparece una persona por superposición, se ha parado en el ultimo punto donde se tiene registro.
- Asumir que la persona sigue manteniendo el mismo vector de movimiento que llevaba antes de desaparecer.

En el caso de activar la primera opción, se incrementa el contador de desapariciones y se mantiene el punto de posición de la persona en el último lugar donde fue detectada.

En el caso de activar la segunda opción, cuando la persona desaparece repentinamente sigue manteniendo el sistema de predicción mediante filtro de Kalman, en base a la velocidad y dirección que llevaba antes de desaparecer. La diferencia es que en este caso únicamente se aplica la predicción y no la corrección. De esta forma si la persona reaparece en la trayectoria que se esperaba podrá volver a ser identificada correctamente.

Tras finalizar todos estos pasos se devolverá la lista actual de las personas identificadas y bajo seguimiento, de vuelta al módulo de detección. La figura 6.7 muestra la estructura general de funcionamiento del módulo de seguimiento.

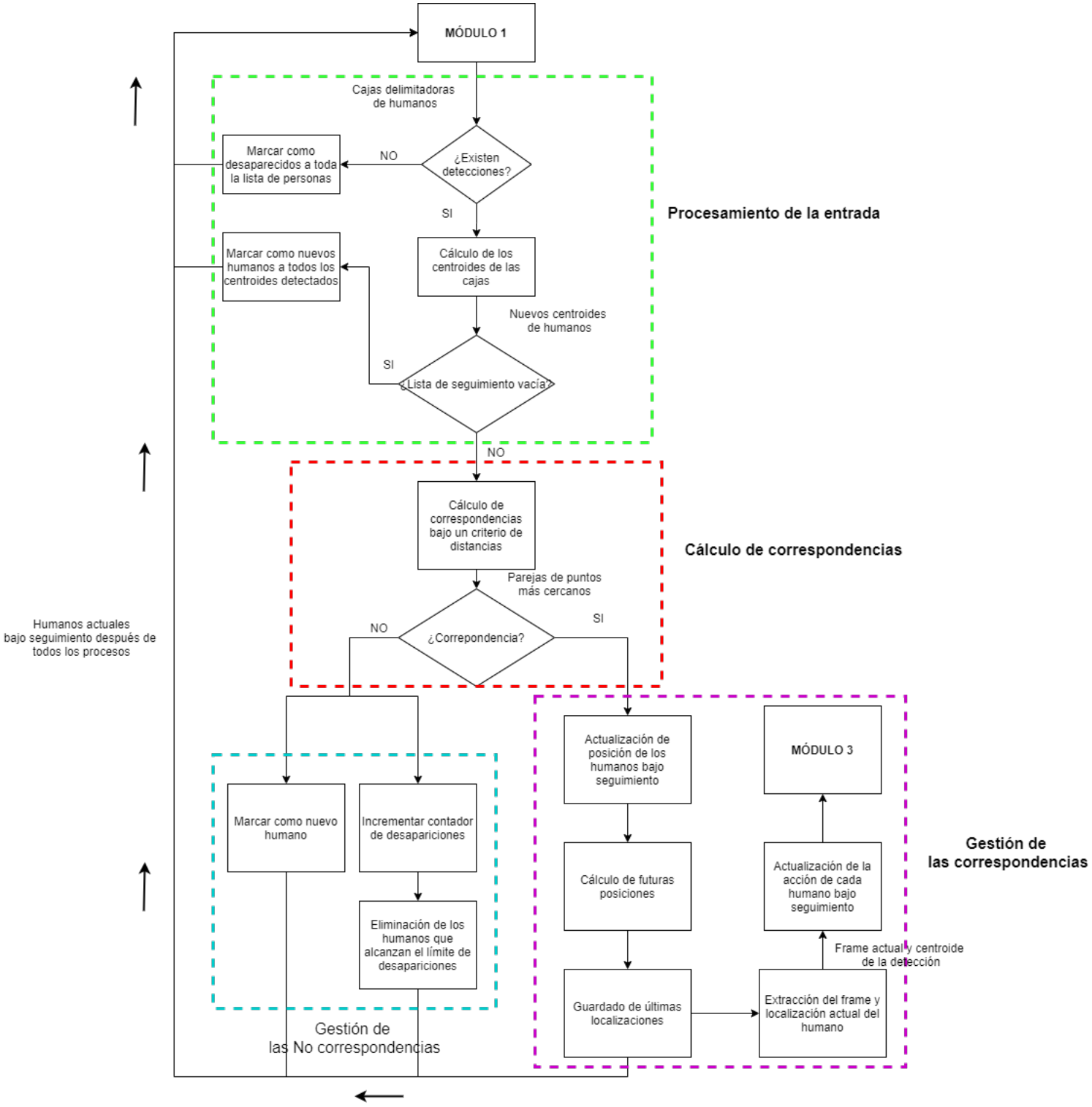

Figura 6.7: Estructura de funcionamiento del módulo de seguimiento

6.4. Módulo de predicción de acciones

Un sistema de videovigilancia inteligente no solo debe ser capaz de detectar e identificar personas, también debe tener la capacidad de detectar qué acciones están realizando estas personas.

Para que un sistema clasifique correctamente la acción realizada por una persona, debe ser capaz de mantener un seguimiento constante de dicha persona, durante al menos un mínimo periodo de tiempo. En este caso, esto se consigue gracias al trabajo previo realizado por el módulo de seguimiento y localización.

Gracias al seguimiento de las personas que lleva el mencionado módulo, el trabajo que debe realizar este módulo, es el de almacenar los *frames* consecutivos del vídeo en los que se detecte esa persona. No obstante, los *frames* deberán ser preprocesados mediante un proceso denominado extracción.

Los procesos que conforman este módulo están divididos de la siguiente forma:

1. **Inicialización de una persona**
   - Inicialización de la lista de *frames*
   - Proceso de extracción
   - Almacenamiento y acción inicial
2. **Actualización de una persona**
   - Comprobaciones previas
   - Método FIFO (First In First Out)
   - Proceso de extracción
   - Almacenamiento y actualización
3. **Predicción de la acción**
   - Comprobación de extracciones
   - Pre-procesado de *frames* extraídos
   - Inferencia sobre el modelo Resnet-3D
   - Procesamiento de la salida de la red

Al contrario que en los anteriores módulos, donde el flujo de ejecución generalmente fluye desde la primera etapa hasta la última, en este caso, cada una de ellas se ejecutará en función de la orden recibida desde el módulo de seguimiento.

No obstante, pese a que la ejecución de las diferentes operaciones se haga de forma independiente dentro del modulo de predicción, su flujo podría definirse en base al orden que tienen las llamadas a dichas operaciones desde el módulo previo que alimenta a éste.

En primer lugar, cuando una persona es añadida a la lista de seguimiento del módulo previo, debe también disponer de la posibilidad de generar una predicción sobre sus acciones. Para lograr esto, el módulo de seguimiento envía a este módulo, tanto el centroide de la persona detectada, como el *frame* en el que se ha producido dicha detección a este módulo.

Lo primero que hace este módulo es crear una lista vacía de imágenes. Esta lista vacía de imágenes será introducida en un diccionario usando como clave el ID generado en el módulo de seguimiento. De esta forma todos los datos relativos a una persona podrán ser accesibles mediante el mismo ID.

Tras crear la lista, el siguiente paso es el de introducir el *frame* en el que se ha detectado a la persona, no obstante, un *frame* integra todo el conjunto de la imagen, en la cuál puede haber muchas otras personas u objetos. Dado que lo único que se busca obtener de la imagen es la persona detectada, se lleva a cabo el proceso de recorte, también denominado extracción.

Este proceso extrae de la imagen exclusivamente la sección de la misma en la que esté la persona. Para ello se establecen durante la inicialización del sistema unas dimensiones fijas de extracción, de esta manera se asegura que la red neuronal recibe una entrada uniforme.

Teniendo en cuenta estas dimensiones fijas, H (altura) y W (anchura). Se persigue cortar una sección de tamaño WxH de la imagen. Estas dimensiones deben ser ajustadas en función de la distancia a la que se sitúe la cámara sobre la escena donde aparecen las personas a detectar, cuanto mas alejada los recortes serán mas pequeños.

En la Figura 6.8 se representa visualmente el proceso de extracción de una persona a partir de un *frame*. En este caso se trata de un recorte totalmente válido en el que no existen problemas de dimensiones ni de posición.

Figura 6.8: Ejemplo de extracción válida

Sin embargo, no siempre es posible llevar a cabo una extracción tan sencilla, dado que mantener unas dimensiones fijas de recorte puede complicar la extracción de aquellas personas situadas cerca de los límites del campo de visión de la cámara.

Cuando los recortes se deben realizar sobre personas cuya posición dista de los bordes horizontales de la imagen una distancia dw, siendo $dw < W/2$, o de los bordes verticales una distancia dh, siendo $dh < H/2$, nos encontramos con problemas de dimensiones. Para solucionar esto, se debe llevar a cabo un método específico, que ha sido denominado, desplazamiento adaptativo de la ventana de extracción.

En la Figura 6.9 se muestra una representación de cómo el desplazamiento adaptativo

de la ventana de extracción, puede solucionar los problemas en los casos citados en el anterior párrafo.

Figura 6.9: Ejemplo de un desplazamiento adaptativo de la ventana de extracción

La utilización de este método soluciona tanto los problemas de recortes erróneos horizontales como verticales. El desplazamiento de la ventana se calcula de la siguiente manera:

- En caso de que $dw < W/2$ por la izquierda: El corte de longitud W se realizará desde el punto $x = 0$ hasta $x = W$.
- En caso de que $dw < W/2$ por la derecha: El corte de longitud W se realizará desde el punto $x = (Ancho_{Frame} - W)$ hasta $x = Ancho_{Frame}$.
- En caso de que $dh < H/2$ por arriba: El corte de longitud H se realizará desde el punto $y = 0$ hasta $y = H$.
- En caso de que $dh < H/2$ por abajo: El corte de longitud H se realizará desde el punto $x = (Alto_{Frame} - H)$ hasta $x = Alto_{Frame}$.

La sección resultante de cortar el *frame* mediante el proceso de extracción se almacena en la lista de s de la persona en cuestión. Ésta posteriormente es almacenada en el

diccionario de personas usando su ID.

En el caso de tratar con una persona ya existente en el sistema, los pasos que se seguirán tendrán como objetivo su actualización. Para ello lo primero que se comprobará de esta persona será si ha llegado al número de *frames* almacenados que se necesitan para realizar una predicción, en el caso de este sistema es 16.

Para aquellas personas cuya lista de *frames* está al máximo, es decir, 16 elementos, se seguirá un método de gestión FIFO(First In First Out). Tal y como se representa en la Figura 6.10, el *frame* más antiguo almacenado será eliminado y se añadirá al final de la cola el nuevo elemento, resultante de aplicar el proceso de extracción sobre el *frame*.

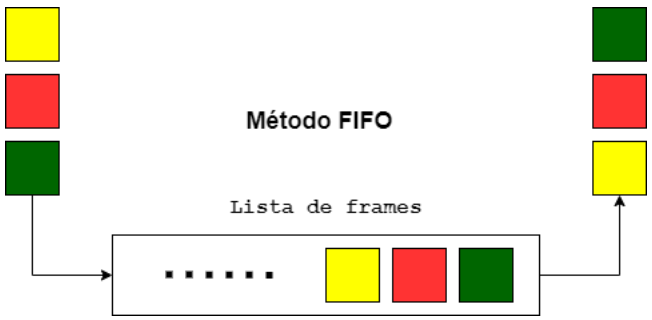

Figura 6.10: Actualización de la lista de *frames* almacenados mediante el método FIFO

Si aún no se han almacenado el número necesario de *frames*, se seguirá un proceso similar al de inicialización, en el cual se realizará el proceso de extracción sobre el *frame* y se almacenará el resultado en la lista de la persona ya existente.

Finalmente la última funcionalidad de este módulo, y la que da su nombre, es la capacidad de clasificar la acción realizada por una persona. Tal y como se explica en los pasos anteriores, una persona tiene asignada una lista de *frames*, no obstante, únicamente aquellas personas cuya lista esté completa podrán recibir una predicción.

Lo primero que realizará el método de predicción será comprobar que el tamaño de la lista de *frames* es el adecuado. En caso de no ser así, se devuelve la predicción genérica: *Por determinar*, pero en el caso en el que la lista sea la adecuada se continuará el proceso.

De la misma manera en la que se preparaban las entradas para la red neuronal de detección de objetos en el apartado 6.2, los *frames* de la lista son pre-procesados y transformados en formato *blob*. El formato *blob* está diseñado con la intención de procesar de la misma forma una colección de imágenes, manteniendo un mismo tamaño, las mismas dimensiones y los mismos canales de color. El procesamiento de los *frames* es el siguiente:

- Dimensionado a 114x114
- Activar la inversión BGR a RGB
- Aplicar *Mean Substraction* o resta de media, en base a la media RGB del dataset de entrenamiento Kinetics 400(34). Los valores medios para cada uno de las tres componentes espectrales son: (R=114.7748, G=107.7354, B=99.4750)

La lista de imágenes resultantes es dada como entrada a la red neuronal.

La idea inicial era trabajar con una red i3D de dos flujos, teniendo como base al trabajo de Carreira et al.(16), mediante la obtención de flujos ópticos. No obstante, pese a que se

llevaron a cabo pruebas con éxito y se implementaron diversos algoritmos de obtención de flujos ópticos, la carga computacional de operar con una red de esa profundidad y complejidad no solo no era asumible para un sistema que intenta abordar tiempos cercanos al procesamiento en tiempo real, sino que no era asumible para las herramientas de trabajo disponibles durante la realización de este proyecto.

Por estas razones, la red utilizada ha sido la red ResNet-34 3D, inspirada en los trabajos de Hara et al.(25) de 2018. En ese trabajo se propone la utilización de redes residuales de tres dimensiones, permitiendo alcanzar grandes precisiones para una tarea como la perseguida en este sistema, sin con ello tener unos costes y tiempos de computación inviables.

De vuelta a la red, ésta procesa la entrada de 16 *frames* concatenados y devuelve las probabilidades predichas para cada una de las acciones posibles. En este caso al estar entrenado con el conjunto de datos Kinetics-400(34), solo hay 400 acciones disponibles.

De entre las probabilidades que se obtienen a la salida, se selecciona aquella con la probabilidad más alta, y se relaciona con la acción correspondiente.

Con la acción ya clasificada, esta información se envía de vuelta al módulo de seguimiento, que la agrupa y la envía al módulo de detección, el cual la dibuja en la pantalla junto a los identificadores y las trayectorias.

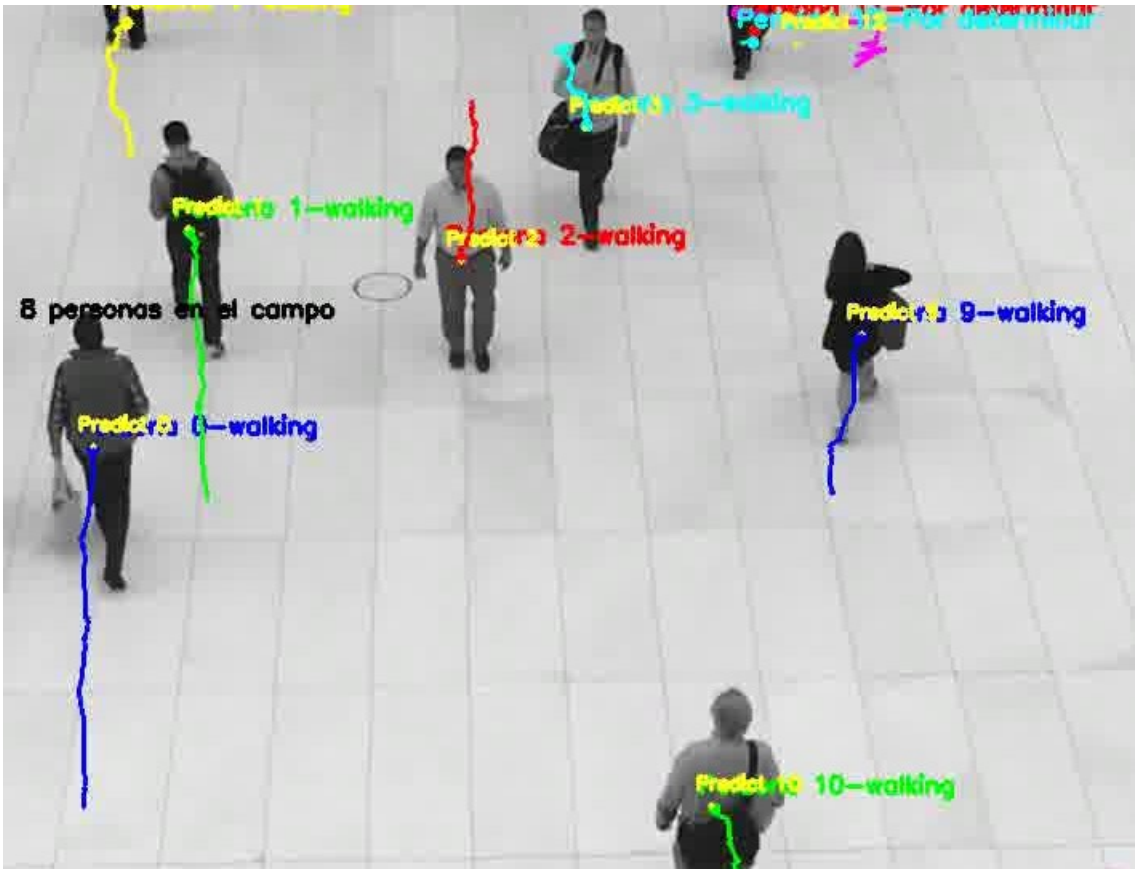

Figura 6.11: Captura del resultado final del sistema

En la figura 6.11 se muestra un ejemplo de funcionamiento de la versión final del sistema, donde aparecen 8 personas. De dos de ellas únicamente se visualizan las piernas, dado que a pesar de que esas personas estén a punto de abandonar el campo de visión de la cámara, el módulo de detección sigue siendo capaz de detectar una persona. En la imagen,todas

las personas muestran sus identificadores únicos, sus trayectorias individuales y su acción clasificada.

La figura 6.12 muestra la estructura general de funcionamiento del módulo de predicción de acciones en humanos.

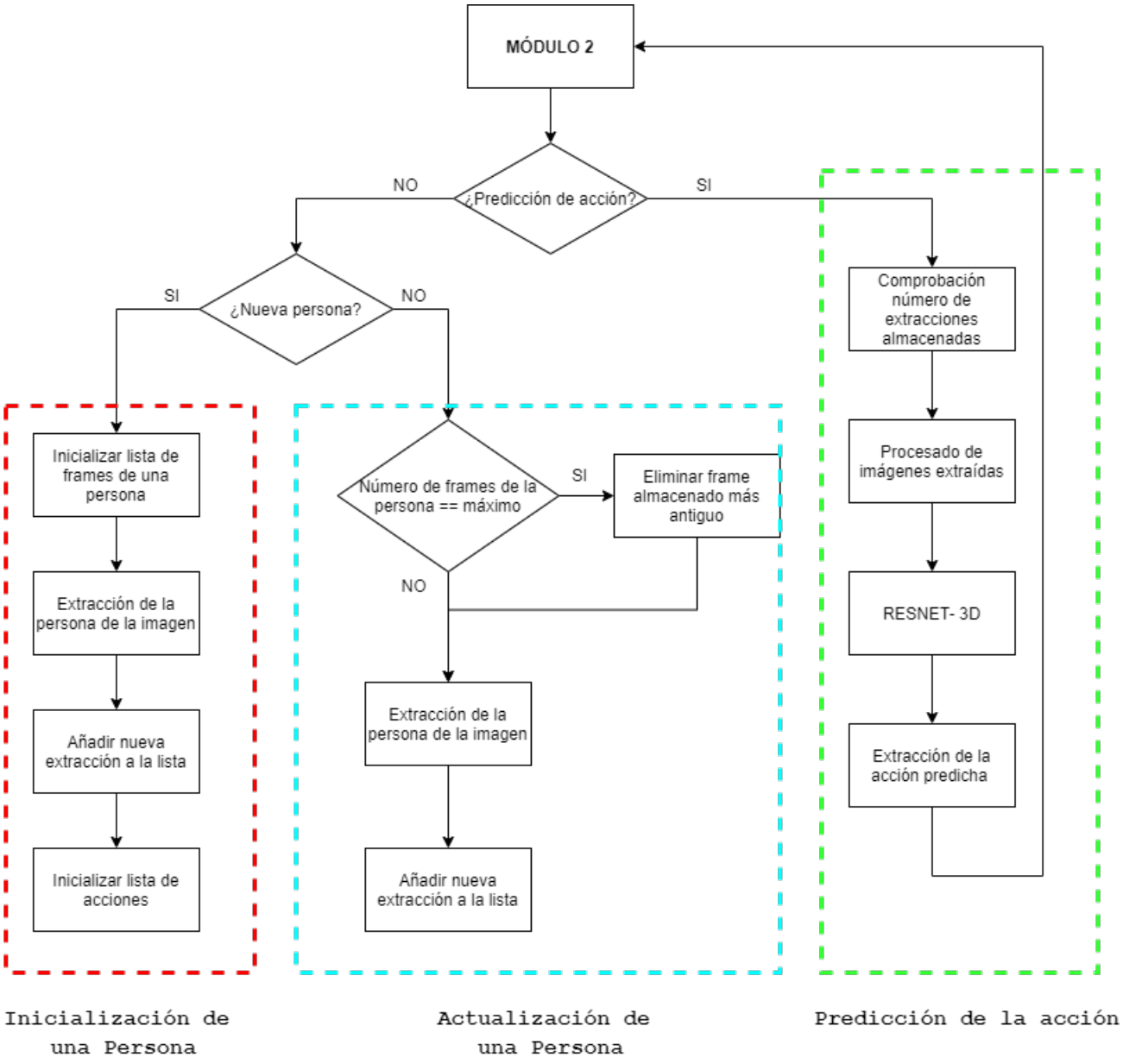

Figura 6.12: Estructura de funcionamiento del módulo de predicción de acciones

El código desarrollado para la aplicación se encuentra en el repositorio TFM bajo licencia MIT(4), mientras que en el Apéndice A se describe el procedimiento de instalación y ejecución.

Capítulo 7

Resultados

7.1. Introducción

Una vez ha finalizado el proceso de diseño e implementación del sistema, llega el momento de probar su funcionamiento en determinadas situaciones, para, de esta manera, detectar sus puntos fuertes y sus debilidades, permitiendo en un futuro realizar mejoras del mismo.

Para ello, se analizan las utilidades del sistema propuesto y la importancia de las funcionalidades ofrecidas. Dada las características del sistema implementado, no existen mecanismos de *benchmark* o medición para este caso concreto; no obstante, caben diversas opciones a la hora de probar el correcto funcionamiento del sistema.

En este caso, se ha optado por realizar una serie de pruebas de funcionamiento con diferentes fragmentos de vídeos cortos y escenarios. Comparando los datos obtenidos por el sistema con los que realmente están ocurriendo en el vídeo es posible comprobar la precisión del sistema. De esta manera las métricas utilizadas a la hora de comprobar la fortaleza del diseño son:

- Número de ID's o cantidad de identificadores únicos otorgados al final de la ejecución.
- Media total del porcentaje de acciones identificadas correctamente en cada *frame*.

No se tomará como métrica la velocidad de procesamiento del sistema, dado que pese a estar diseñado con la intención de alcanzar velocidades de procesamiento en tiempo real, la capacidad de cómputo de la máquina disponible para la realización de las pruebas de funcionamiento no es la adecuada para un sistema de dichas características.

Concretamente, la máquina donde se ha implementado el sistema posee las siguientes especificaciones: Intel Core i5, RAM 6 GB, Sistema Operativo Ubuntu y ausencia de GPU. Por tanto, estas características permiten establecer un límite inferior por debajo del cual el funcionamiento en tiempo real no es factible.

Al tratarse de un sistema diseñado con el objetivo de realizar labores de videovigilancia inteligente, todos los casos de uso tendrán como base la utilización de vídeos de cámaras fijas de videovigilancia, que proporcionan los vídeos utilizados.

A la hora de inicializar el sistema, pueden adaptarse una serie de parámetros en función

del tipo de escenario donde este sistema se vaya a implantar, de forma que las métricas finales del sistema pueden variar según se utilicen unos valores u otros.

Estos parámetros son:

1. *Confianza*: mínimo nivel de umbral en una detección.

2. *Max Desaparición*: número máximo de *frames* continuos en los que una persona puede no ser detectada sin ser eliminada.

3. *Max Distancia*: distancia máxima existente entre una pareja con relación de correspondencia para considerar que puedan ser la misma persona.

4. *Tam Acción*: número de *frames* necesarios de una persona para poder realizar una predicción sobre la acción que realiza.

7.2. Vídeos de pruebas

Para la realización de las pruebas presentadas en esta memoria se han utilizado fragmentos cortos de vídeos correspondientes a secciones recortadas de vídeos de la plataforma *Pexels* (6), los cuales pueden ser utilizados libremente bajo licencia gratuita.

Estos vídeos muestran grabaciones de una cámara de seguridad en una calle durante varios segundos, lo que los hace idóneos para llevar a cabo este formato de prueba, ya que el uso de fragmentos de corta duración en el sistema no afecta al cómputo de las métricas, dado que su eficiencia no depende de la longitud de un vídeo.

Se han utilizado en total 60 recortes de vídeo procedentes de la mencionada plataforma. Dichos vídeos poseen especificaciones técnicas comunes: duración promedio 1 segundo, capturados a razón de 30 *frames* por segundo aproximadamente y por tanto en cada secuencia se utilizan del orden de 30 *frames* en el modelo de color RGB con 24 bits por píxel.

Las resoluciones espaciales (ancho y alto) de cada *frame* varían según los vídeos, si bien en el caso de las utilizadas en las pruebas es de 500x480 píxeles. Los vídeos completos en su totalidad tienen duraciones variables siempre superiores a la duración de los recortes utilizados en las pruebas, superando con carácter general los 40 segundos.

Del conjunto de vídeos seleccionados se han identificado dos tipos de vídeos, que se utilizan, con carácter didáctico, como base de las pruebas realizadas. Los vídeos del primer tipo, conteniendo 40 recortes, han sido grabados por una cámara en un ángulo de visión con una panorámica elevada que minimiza el efecto de oclusiones de personas y bajo condiciones lumínicas ideales. En el segundo tipo de vídeos, conteniendo los 20 recortes restantes, la cámara posee un peor ángulo de vista que favorece la pérdida de registro de personas, provocando fallos debido al solapamiento entre las personas, que tal y como se describe en la sección 8.2, podrían ser solucionados utilizando dos cámaras situadas en diferente posición y ángulo. A continuación se analizan los resultados de estos dos tipos de vídeos.

7.3. Análisis de resultados de los tipos de vídeo

7.3.1. Tipo de vídeos 1

Para el análisis de este tipo de vídeos se han definido los valores de los parámetros indicados, tal y como se han definido previamente. Cuyos resultados promedios obtenidos con los 40 vídeos utilizados son los que se especifican en relación al número de IDs generados y el porcentaje de aciertos. Para establecer estos valores se ha utilizado el criterio de un supervisor humano, de forma que es quien establece, en base a su criterio y experiencia, el resultado de la medición, ya que como se ha indicado previamente, no existe un *benchmark* al respecto.

Parámetros	Confianza	Max Desaparición	Max Distancia	Tam Acción
Valores	0.6	20 *frames*	70 px	16 *frames*

Métricas	Número de ID's generadas	Porcentaje de acierto de acciones
Resultado	12	75 %

La figura 7.1 muestra un fragmento representativo correspondiente a uno de los vídeos de tipo 1. En él se observa que todas las personas del vídeo han sido identificadas correctamente, han recibido su ID único y su acción ha sido clasificada también correctamente. Al tratarse de una cámara de vigilancia en una zona de tránsito, es de esperar que las personas aparezcan caminando, como puede demostrar el hecho de que la mayoría hayan sido clasificadas como *walking*. Sin embargo, en el caso del ID 9, ha sido clasificado como *shoveling snow* (retirando nieve), esto es debido a la tonalidad blanca del suelo y a que el modelo de predicción de acciones ha sido entrenado con un *dataset* genérico de acciones (*Kinetics-400*), tal y como se ha indicado previamente, no en uno especializado para peatones.

Si bien no todas las acciones de las personas en el vídeo se clasifican correctamente, las predicciones erróneas hacen referencia a acciones que implican movimientos similares al original.

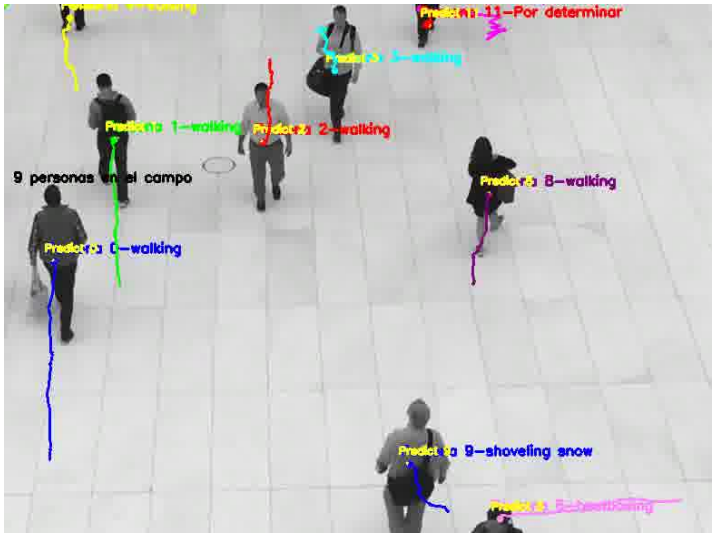

Figura 7.1: *Frame* procesado por el sistema, procedente de un vídeo tipo 1.

7.3.2. Tipo de vídeos 2

Para el análisis de este segundo tipo de vídeos, se utilizan los parámetros que se especifican a continuación para computar los valores de las métricas utilizadas, que determinan igualmente el número de IDs generadas y el porcentaje de aciertos de las acciones, utilizando el mismo criterio de supervisión del experto.

Parámetros	Confianza	Max Desaparición	Max Distancia	Tam Acción
Valores	0.6	20 *frames*	70 px	16 *frames*

Métricas	Número de ID's generadas	Porcentaje de acierto de acciones
Resultado	7	65 %

Al contrario que en la prueba uno, en el vídeo utilizado en esta prueba la cámara tiene un ángulo de perspectiva demasiado bajo, lo que provoca que el algoritmo de detección no sea capaz de determinar la posición de todas las personas en todo momento. Debido a este mismo problema de ángulo, se producen también problemas de solapamiento durante largos periodos de tiempo entre las personas, provocando en el ejemplo del *frame* que se muestra en la figura 7.2 que una de ellas pierda varias veces su identificador, teniendo éste que ser reiniciado otras tantas veces.

La pérdida de identificador implica que si posteriormente el algoritmo de detección es capaz de detectar a la persona perdida previamente, ahora la considere como una nueva, asignándola otro ID.

La figura 7.2 muestra un fragmento representativo correspondiente a uno de los vídeos de tipo 2. Como puede observarse solo se detectan tres personas en el campo, a pesar de que existen cuatro. Una de ellas ha sido solapada tras la primera durante un largo número de *frames*, provocando la pérdida de su identificador. De esas personas, dos han sido predichas correctamente al arrastrar delante un objeto que el sistema ha entendido como un carro, asignando la acción *pushing car*, en cambio la tercera ha sido clasificada como *jogging* o haciendo yoga. Al igual que en el caso de la prueba uno, esto es debido a la poca especialización del *dataset*, que no está convenientemente adaptado a situaciones peatonales.

Figura 7.2: *Frame* procesado por el sistema, procedente de un vídeo tipo 2.

Capítulo 8

Conclusiones y trabajo futuro

8.1. Conclusiones

Tras la finalización del diseño del sistema y su desarrollo a lo largo del presente proyecto, llega el momento de realizar una valoración sobre los objetivos cumplidos. En base a las pruebas realizadas sobre el sistema final, se puede afirmar que los objetivos impuestos durante la especificación del proyecto se han cumplido satisfactoriamente.

En primer lugar, el sistema implementado puede detectar personas en imágenes, siendo capaz de recibir vídeos, muestrearlos en *frames* y aislar las posiciones de las personas presentes en ellos, marcándolas de una forma precisa.

Gracias a la anterior funcionalidad, el sistema adquiere la capacidad de analizar las coordenadas resultantes de las detecciones y predecir dónde se encontrarán estas mismas personas en un futuro, e identificarlas de manera única e independiente. Además, tiene capacidad de gestionar el ciclo de vida del paso de una persona por el campo de visión de la cámara, almacenando sus trayectorias de movimiento y resaltándolas sobre cada *frame* del vídeo.

Además, el sistema hace uso de esta capacidad de seguimiento e identificación única, para extraer las secciones de un *frame* donde la persona está presente. El agrupamiento y procesado de estas secciones permite extraer y clasificar la acción que está realizando la persona en el espacio temporal definido por ese conjunto de *frames*.

La integración de estos módulos, todos ellos con funcionalidades diferenciadas, constituyen el sistema de videovigilancia inteligente, que es el objetivo principal del proyecto.

Durante el desarrollo de este proyecto han surgido una serie de dificultades a las que se ha debido hacer frente para completar con éxito el proyecto propuesto.

En un inicio, el objeto de estudio como aspirante a modelo de redes neuronales en el módulo de predicción de acciones, fueron las redes i3D(16), en una arquitectura de dos flujos, incluyendo el procesamiento de flujos ópticos. Sin embargo, pese a la realización de diversas variantes de procesamiento de flujos ópticos, este tipo de red tan profunda demostró ser demasiado lenta para un sistema que aspira a realizar procesamiento en tiempo real, necesitando además de una capacidad de cómputo que no estaba disponible durante la realización de este trabajo. De esta forma el nuevo objeto de estudio fueron las redes Resnet-3D, publicadas por Hara et al.(25). Su incorporación supuso un gran acierto,

al permitir realizar predicciones a velocidades mucho más altas sin con ello reducir de manera significativa la precisión.

En resumen, en este proyecto se han investigado y puesto en común diferentes ramas de la visión por computadora y del aprendizaje automático, culminando en la implementación de un sistema de videovigilancia inteligente. No obstante, la implementación del sistema no era el único objetivo, otra serie de propósitos ligados al desarrollo del proyecto también se han visto satisfechos, en especial la oportunidad de haber podido ampliar considerablemente mis conocimientos en ciertas áreas de la visión por computadora y del aprendizaje profundo.

8.2. Trabajo futuro

Si bien el sistema implementado ha funcionado correctamente en el entorno ante el que se ha probado, existen algunas situaciones en las que su funcionamiento podría no ser el esperado. En este caso, uno de los problemas que pueden ocurrir son los relacionados con la superposición de cajas de detección entre personas.

Este suceso, pese a estar parcialmente solucionado gracias a la predicción de los movimientos futuros de las personas en el módulo de seguimiento, puede cometer errores de identificación en aquellos casos en los que el ángulo de visión de la cámara no sea el adecuado.

Una manera de corregir esto, es la ampliación del sistema para trabajar con dos cámaras. Cada una de ellas con un punto de vista diferente, de manera que se pueda crear un mapa en tres dimensiones y en profundidad del escenario bajo vigilancia, evitando aquellos casos en los que dos elementos del plano se superpongan de manera inevitable al no tener concepto de profundidad.

Otra vertiente de trabajo futuro sería la elaboración de un *dataset* especializado en acciones de peatones, de manera que el sistema pueda alcanzar mayores precisiones en este ámbito. Así como la implementación de un sistema de procesamiento paralelo en el módulo de predicción, permitiendo aumentar la velocidad de respuesta ante grandes cantidades de personas.

Este campo de estudio se encuentra en continuo crecimiento y uno de los grandes puntos de trabajo futuro del sistema, seria la investigación e integración de los nuevos algoritmos de predicción de objetos y de detección de objetos que van apareciendo, como por ejemplo el nuevo modelo de detección de objetos Yolov5(7), publicado apenas unas semanas antes de la finalización de esta memoria.

Chapter 9

Introduction

"Some people call this technology artificial intelligence, when in fact what it will allow is to increase our own"
— Gin Rometty

9.1. Motivation

We, as human beings, have always tried to obtain as much information as possible about the environment around us, either directly or indirectly. In this way the human being has always sought sources of information, from which to learn, from which to take advantage. But it has not been until the advances in computation of the last decades, that these tasks of obtaining and analyzing information have been, to some extent, assigned to the machines.

To be fair, the speed of sequential human reasoning is low compared to computers, so it can be thought that any problem expressed in the form of an algorithm, and given to a computer could always be solved. However there is something in which the human being surpasses the machine, the possibility of making judgments and obtaining information within contexts of which he knows nothing previously.

On the contrary, machines require a lot of information to train, especially if we apply deep learning. But once our machine has learned, it can extract knowledge from this information, in this case the activities carried out in a video, more accurately than a human.

The possibility of detecting what action or activity is being developed in a video, feeds on the motivation arising from thinking about the uses that could be given. For example, being able to tag videos automatically would be very useful to speed up searches in video databases, also for example detecting a baseball bat, someone playing the guitar or someone shaving. Its application in automatic video surveillance is an aspect with a wide range of possibilities, both to detect criminal activities and even be able to predict them.

This is where deep learning comes in, which together with computer vision techniques, allow us to detect different activities from sequences of images (videos).

9.2. Objectives

Although, the final objective of this work project is:

- Design and implement a video surveillance system capable of detecting humans, tracking them as independent entities, predicting their movements and recognizing the actions they are doing.

The intermediate objectives requiered to achieve it could be divided into several groups:

- Investigate the techniques used in video recognition, implementing in particular the architecture of two flows of i3D convolutional networks and the Resnet 3D arquitecture.
- Investigate and implement several techniques to obtain optical flows.
- Investigate and use the most advanced techniques for detecting objects in images, seeking to prioritize computing speed in real time.
- Implement a human tracking system within a video plane, capable of tracking their past movements and predicting future ones.

The tasks addressed during the development of the work could be classified within the following fields of computer science:

- **Artificial Inteligent**, specifically the branch of deep learning, inside machine learning.
- **Computer Vision**, regarding the techniques for computing optical flows in temporal sequences of images and the image processing.

9.3. Workplan

The implementation of this project was initially divided into the following phases, however, it has not always been possible to strictly follow this plan, and under certain circumstances they have been subject to change. The phases are:

- Phase 0: Definition of the system to be carried out and the necessary parts that integrate it.
- Phase 1: Investigation of techniques and algorithms for detecting objects in videos.
- Phase 2: Design and implementation of a system capable of detecting humans in videos, according to the needs of the project.
- Phase 3: Investigation of techniques and algorithms to track a large number of independent entities within a video, according to the needs of the project.
- Phase 4: Design and implementation of a video tracking system for humans, capable of assigning unique identifiers to each of the humans, storing their past movements and predicting their future movements.

- Phase 5: Investigation of techniques and algorithms for recognizing human actions in videos.
- Phase 6: Design and implementation of a system capable of recognizing the human actions present in videos, prioritizing speed in real time.
- Phase 7: Integration of the components previously developed with the aim of forming the intelligent video surveillance system
- Phase 8: Carrying out tests on the final system.
- Phase 9: Writing the project report.

The mentioned phases intrinsically constitute the main personal contribution to the work carried out, highlighting the study of techniques related to detecting people in videos and the application of deep learning techniques, in order to identify the most appropriate ones, to carry out the design and integration of them in a system, which allows the analysis of results.

9.4. Memory organization

A continuation expongo brevemente los contenidos de cada capitulo que conforma la memoria.

- **Chapter 1: Introduction**

  It is the introductory chapter to the project, which describes the motivation behind the development of this work, the objectives that are sought during the resolution of the same, as well as the work plan to be followed and the organization of memory.

- **Chapter 2: Previous work**

  Throughout this chapter the most relevant aspects of the investigation of the state of the art, of the existing techniques in the field of recognition of human activities in videos, from their origin to the present, are described. Finally, some of the technologies investigated for the development of the project are exposed.

- **Chapter 3: Optical flows**

  Entering the field of artificial vision, this chapter analyzes the theory of movement in its psychological variant, in an introductory way to be able to understand below the concepts on which the optical flow of a video is based.

- **Chapter 4: Artificial neural networks**

  In this chapter we enter the world of neural networks, starting from a biological approach, to gradually understand the concepts inherent to them.The main focus is on the convolutional neural networks, and on the layers which form them, however, they also describe other types of networks that have also been used in this branch.

- **Chapter 5: Working methodology**

  In this section I explain the methodologies followed during the realization of the project, as well as some of the tools that have been useful for organizing the work.

- **Chapter 6: Activity recognition system**

  The main purpose of this chapter is to detail how the system developed has been built, explaining its general operation and the components that comprise it.

- **Chapter 7: Results**

  In this chapter the results obtained from performing different tests are collected in the form of tables.

- **Chapter 8: Conclusions and future work**

  This section contains the conclusions obtained after reflecting the results of the tests carried out, as well as taking into account possible future work that could be carried out.

Chapter 10

Conclusions and future work

10.1. Conclusions

After completing the design and development of the proposed system throughout the project, it is time to make an assessment of the objectives achieved. Based on the tests carried out on the final system, it can be concluded that the objectives fixed during the project specification have been satisfactorily fulfilled.

Firstly, the implemented system can detect people in images, being able to receive videos, sample them in frames and isolate the positions of the people present in them, marking them accurately.

Thanks to the previous functionality, the system acquires the ability to analyze the coordinates resulting from the detections and predict where these same people will be found in the future, and identify them in a unique and independent way. In addition, it will manage the life cycle of a person's passage through the camera's field of vision, storing it's movement trajectories and highlighting them on the video.

In addition, the system makes use of this unique tracking and identification capacity, to extract the sections of a frame where the person is present. The grouping and processing of these sections allows to extract and classify the action that the person is performing in the time space defined by that set of frames.

The integration of these modules, all of them with differentiated functionalities, constitute the intelligent video surveillance system, which is the main objective of the project.

During the development of this project, a number of difficulties have arisen which have had to be faced in order to successfully complete the proposed project.

Initially, the object of study as a candidate for a neural network model in the action prediction module was i3D networks (16), in a two-flow architecture, including the processing of optical flows. However, despite the implementation of several variants of optical flow processing, this type of deep network proved to be too slow for a system that aspires to perform real-time processing, also requiring computing capacity that was not available during the development of this work. Thus, the new object of study was Resnet-3D networks, published by Hara et al.(25). It was a great success, allowing predictions to be made at much higher speeds without significantly reducing accuracy.

In summary, in this project different branches of computer vision and deep learning have been investigated and integrated, culminating in the implementation of an intelligent video surveillance system. However, the implementation of the system was not the only objective, other series of purposes linked to the development of the project have also been met, especially the opportunity to have been able to considerably expand my knowledge in certain areas of computer vision and deep learning.

10.2. Future work

Although the implemented system has provided satisfactory results in the environment where it has been tested, there are some situations in which its operation may not be as expected. In this case, one of the problems that can occur is related to the overlapping of detecting bounding boxes between people.

This event, despite being partially solved thanks to the prediction of the future movements of people in the tracking module, can make identification of errors in those cases in which the camera's viewing angle is not adequate.

One way to correct this is to expand the system to work with two cameras. Each of them with a different point of view, so that a three-dimensional and in-depth map of the scene can be created under surveillance, avoiding those cases in which two elements of the plane inevitably overlap due to the lack of concept of depth.

Another aspect of work would be the elaboration of a dataset specialized in pedestrian actions, so that the system can achieve greater precision in this area.As well as the implementation of a parallel processing system in the prediction module, allowing to increase the speed of response to large numbers of people.

This field of study is in continuous growth and one of the great future work points of the system, would be the research and integration of new algorithms for action prediction and object detection, such as the new object detection model Yolov5(7), published just a few weeks before the end of this report.

Apéndice A

Ejecución del sistema

En este capítulo se explican los aspectos de interés a la hora de instalar y utilizar el sistema implementado. En primer lugar se indica de nuevo la dirección del repositorio desde el cual se puede acceder al código del proyecto.

- Enlace al proyecto: Repositorio.

La estructuración de carpetas del sistema en su implementación final se organiza de la siguiente forma:

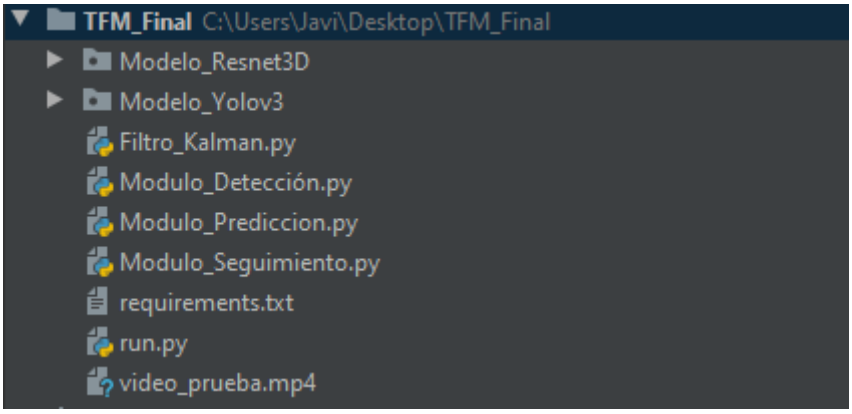

Figura A.1: Estructura de carpetas del sistema implementado

- **Modelo Resnet 3D**: este directorio almacena todos los datos necesarios para utilizar el modelo Resnet-34 3D diseñado por Hara et al(25). En concreto el propio modelo preentrenado en Kinetics-400 almacenado en formato *.onnx* y las etiquetas del propio *dataset*.
- **Modelo Yolov3**: esta carpeta almacena el modelo Yolov3 diseñado por Redmon et al(48) y los pesos resultantes de su entrenamiento con el *dataset* COCO(39). Incluye también las etiquetas del *dataset* mencionado.
- **Filtro Kalman.py**: en este *script* se halla la implementación usada por el sistema del algoritmo del Filtro de Kalman (32).

- **Módulo Detección.py**: contiene el código que implementa las funcionalidades descritas en la sección 6.2 para el módulo de detección de objetos.
- **Módulo Predicción.py**: este archivo de código implementa las funcionalidades descritas en la sección 6.4 para el módulo de predicción de acciones humanas.
- **Modulo Seguimiento**: en este archivo se encuentra el código que implementa las funcionalidades descritas en la sección 6.3 para el módulo de seguimiento de personas.
- **requirements.txt**: este archivo contiene todas las librerías y dependencias necesarias para que el sistema funcione correctamente.
- **run.py**: este *script* arranca el sistema mediante una serie de parámetros que permitirán personalizar su funcionamiento.
- **vídeo de prueba.mp4**: vídeo de prueba para realizar una ejecución.

Una vez queda clara la estructura de carpetas del sistema, se deben seguir una serie de pasos para poder ejecutar el sistema correctamente:

1. Descargar el repositorio en local y situarse dentro de él.
2. Crear un entorno virtual con Python 3.6.4 en la máquina y activarlo.
3. Instalar las dependencias y librerías utilizadas por el sistema mediante el comando: "*pip install -r requirements.txt*"
4. Una vez todo se ha instalado correctamente, ejecutar el sistema mediante el comando: "*python run.py* "

   Este comando admite hasta nueve parámetros diferentes, cuyas nomenclaturas se encuentran expresadas dentro del propio archivo *run.py*.
5. Para detener la ejecución mantener pulsada la tecla **p**.

Para ejecutar el sistema utilizando diferentes parámetros a los que tiene por defecto, se debe acceder al *script* run.py, donde pueden ser modificados los valores de los siguientes parámetros:

- **CONFIANZA**: umbral para la detección de objetos.
- **MAX DESAPARICION**: número máximo permitido de desapariciones.
- **MAX TRAZA**: longitud de la trayectoria dibujada.
- **MAX DISTANCIA**: distancia máxima para tolerar correspondencias.
- **TAM**: tamaño TAMxTAM de la ventana de extracción.
- **ACTION DURATION**: número de *frames* necesarios para calcular la acción.

Para poder modificar mas fácilmente los atributos utilizados durante una ejecución, los parámetros listados anteriormente pueden ser modificados directamente desde el comando de ejecución del sistema, tal y cómo aparece representado dentro del archivo *run.py*.

Bibliografía

[1] GitHub: Guides. https://guides.github.com/. Accessed: 2020-04-13.

[2] Numpy. https://numpy.org/doc/stable/. Accessed: 2020-03-28.

[3] Open CV. https://opencv.org/about/. Accessed: 2020-03-27.

[4] Open Source.org: Mit license. https://opensource.org/licenses/MIT. Accessed: 2020-06-31.

[5] Pandas. https://pandas.pydata.org/docs/. Accessed: 2020-03-28.

[6] Pexels: free stock photos. https://www.pexels.com/es-es/. Accessed: 2020-05-20.

[7] Toward Data Science: Yolov5 is here! https://towardsdatascience.com/yolo-v5-is-here-b668ce2a4908. Accessed: 2020-06-30.

[8] Wikipedia: Google Drive. https://es.wikipedia.org/wiki/Google_Drive. Accessed: 2020-05-17.

[9] Wikipedia: RGB color model. https://en.wikipedia.org/wiki/RGB_color_model. Accessed: 2020-02-18.

[10] Wikipedia: Ruby on Rails. https://es.wikipedia.org/wiki/Ruby_on_Rails. Accessed: 2020-05-20.

[11] ¿Qué es Trello y cómo se usa? https://trello.com/b/FYapG0ok/ayuda-%E2%86%92-%E2%86%92-unite-%E2%86%92-%E2%86%92-metodolog%C3%ADa. Accessed: 2020-04-02.

[12] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. J. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Józefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. G. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. A. Tucker, V. Vanhoucke, V. Vasudevan, F. B. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *ArXiv*, abs/1603.04467, 2016.

[13] B. F. Buxton and H. Buxton. Computation of optic flow from the motion of edge features in image sequences. *Image Vis. Comput.*, 2:59–75, 1984.

[14] J. Carreira, E. Noland, A. Banki-Horvath, C. Hillier, and A. Zisserman. A short note about kinetics-600. *ArXiv*, abs/1808.01340, 2018.

[15] J. Carreira, E. Noland, C. Hillier, and A. Zisserman. A short note on the kinetics-700 human action dataset. *ArXiv*, abs/1907.06987, 2019.

[16] J. Carreira and A. Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4724–4733, 2017.

[17] R. Chaudhry, A. Ravichandran, G. Hager, and R. Vidal. Histograms of oriented optical flow and binet-cauchy kernels on nonlinear dynamical systems for the recognition of human actions. *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[18] C.-C. Chen and J. K. Aggarwal. Recognizing human action from a far field of view. *2009 Workshop on Motion and Video Computing (WMVC)*, 2009.

[19] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20:273–297, 2004.

[20] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR05)*, 2015.

[21] G. Farnebäck. Two-frame motion estimation based on polynomial expansion. In *SCIA*, 2003.

[22] J. J. Gibson. The perception of the visual world. 1950.

[23] R. B. Girshick. Fast r-cnn. *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1440–1448, 2015.

[24] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 580–587, 2014.

[25] K. Hara, H. Kataoka, and Y. Satoh. Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet? *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6546–6555, 2018.

[26] C. G. Harris and M. Stephens. A combined corner and edge detector. In *Alvey Vision Conference*, 1988.

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

[28] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9:1735–1780, 1997.

[29] B. K. P. Horn and B. G. Schunck. Determining optical flow. *Artif. Intell.*, 17:185–203, 1981.

[30] G. Huang, Z. Liu, and K. Q. Weinberger. Densely connected convolutional networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2261–2269, 2017.

[31] G.-B. Huang. What are extreme learning machines? filling the gap between frank rosenblatt's dream and john von neumann's puzzle. *Cognitive Computation*, 7(3):263–278, 2015.

[32] R. E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[33] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[34] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, A. Natsev, M. Suleyman, and A. Zisserman. The kinetics human action video dataset. *ArXiv*, abs/1705.06950, 2017.

[35] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[36] H. W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2:83–97, 1955.

[37] S. S. Kumar and M. John. Human activity recognition using optical flow based feature set. *2016 IEEE International Carnahan Conference on Security Technology (ICCST)*, 2016.

[38] K. Lertniphonphan, S. Aramvith, and T. H. Chalidabhongse. Human action recognition using direction histograms of optical flow. *2011 11th International Symposium on Communications Information Technologies (ISCIT)*, 2011.

[39] T.-Y. Lin, M. Maire, S. J. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. *ArXiv*, abs/1405.0312, 2014.

[40] T. Linus. Git. https://git-scm.com/. Accessed: 2020-04-10.

[41] B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision jcai). 1981.

[42] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. 1967.

[43] W. S. McCulloch and W. F. Pitts. A logical calculus of the ideas immanent in nervous activity. 1990.

[44] O. Oreifej and Z. Liu. Hon4d: Histogram of oriented 4d normals for activity recognition from depth sequences. *2013 IEEE Conference on Computer Vision and Pattern Recognition*, 2013.

[45] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch:

An imperative style, high-performance deep learning library. *ArXiv*, abs/1912.01703, 2019.

[46] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016.

[47] J. Redmon and A. Farhadi. Yolo9000: Better, faster, stronger. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6517–6525, 2017.

[48] J. Redmon and A. Farhadi. Yolov3: An incremental improvement. *ArXiv*, abs/1804.02767, 2018.

[49] S. Ren, K. He, R. B. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39:1137–1149, 2015.

[50] I. Rodríguez-Moreno, J. M. Martínez-Otzeta, B. Sierra, I. Rodriguez, and E. Jauregi. Video activity recognition: State-of-the-art. *Sensors*, 19(14):3160, 2019.

[51] F. F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65 6:386–408, 1958.

[52] G. V. Rossum. Python. *Handbook of Object Technology*, 1998.

[53] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[54] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and L. Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115:211–252, 2015.

[55] S. Satyamurthi, J. Tian, and M. C. H. Chua. Action recognition using multi-directional projected depth motion maps. *Journal of Ambient Intelligence and Humanized Computing*, 2018.

[56] C. Schuldt, I. Laptev, and B. Caputo. Recognizing human actions: a local svm approach. *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, 2004.

[57] S. Sehgal. Human activity recognition using bpnn classifier on hog features. *2018 International Conference on Intelligent Circuits and Systems (ICICS)*, 2018.

[58] J. Shi and C. Tomasi. Good features to track. *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600, 1994.

[59] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *NIPS*, 2014.

[60] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2015.

[61] K. Soomro, A. R. Zamir, and M. Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *ArXiv*, abs/1212.0402, 2012.

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

[63] D. Tran, L. D. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 4489–4497, 2015.

[64] 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*, 6:1155–1166, 2018.

[65] L. Wang, Y. Xiong, Z. Wang, and Y. Qiao. Towards good practices for very deep two-stream convnets. *ArXiv*, abs/1507.02159, 2015.

[66] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. V. Gool. Temporal segment networks: Towards good practices for deep action recognition. In *ECCV*, 2016.

[67] X. Yang, C. Zhang, and Y. Tian. Recognizing actions using depth motion maps-based histograms of oriented gradients. *Proceedings of the 20th ACM international conference on Multimedia - MM 12*, 2012.

[68] C. Zach, T. Pock, and H. Bischof. A duality based approach for realtime tv-l1 optical flow. In *DAGM-Symposium*, 2007.