Departamento de
Informática e Ingeniería
de Sistemas
Universidad Zaragoza
1542

Control de aforos mediante sistema multi-cámara

Francisco Morés Abad

Directora: Ana Cristina Murillo

Ingeniería Informática
Computación

Departamento de Informática e Ingeniería de Sistemas
Escuela de Ingeniería y Arquitectura
Universidad de Zaragoza

Fecha 21 de Junio de 2021

TRABAJOS DE FIN DE GRADO / FIN DE MÁSTER

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe entregarse en la Secretaría de la EINA, dentro del plazo de depósito del TFG/TFM para su evaluación).

D./Dª. Francisco Morés Abad ,en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza, Declaro que el presente Trabajo de Fin de (Grado/Máster) Grado (Título del Trabajo)

Control de aforos mediante sistema multi-cámara

es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada debidamente.

Zaragoza, 21 de Junio de 2021

Fdo: Francisco Morés Abad

Resumen

Los problemas de detección y re-identificación de personas han ido ganando interés en la ultima década y esto se puede apreciar con la gran diversidad de algoritmos y modelos novedosos que han ido apareciendo con el paso del tiempo. Algunos problemas en auge hoy en día, como sistemas de vídeo-vigilancia inteligentes o el control de aforos, son ejemplos de situaciones en las que son necesarias la detección y re-identificación de personas.

Este trabajo aborda el problema del control de aforos mediante un sistema de múltiples cámaras, cuyo reto principal es la integración de varias tareas: lectura y puesta en común de la información de sensores de visión, detección de personas sobre los datos recibidos, re-identificación de las personas detectadas y visualización de resultados. La re-identificación no consiste en identificar (reconocer) a cada individuo, sino simplemente saber cuándo dos imágenes corresponden a distintos puntos de vista de la misma persona, o no. Existen multitud de trabajos que se centran en la detección y/o re-identificación de personas y existen diversas herramientas para lograrlo. En este proyecto se busca diseñar un sistema que se abstraiga de la posición, oclusión o número de cámaras utilizadas con el fin de conseguir la mayor flexibilidad para su uso en situaciones reales. Por lo tanto, el objetivo general es diseñar e implementar un sistema que pueda contar el aforo de una escena monitorizada por varias cámaras.

Tras un estudio de las diferentes técnicas de detección y re-identificación que componen el estado del arte, se diseñó el sistema desarrollado, que consta de los siguientes módulos: un módulo de detección encargado de detectar a las personas pertenecientes a un instante de tiempo, un módulo de re-identificación encargado de emparejar las personas detectadas entre las imágenes de las diversas cámaras, y un módulo de evaluación y visualización encargado de evaluar el sistema (si los datos utilizados poseen anotaciones) y plasmar los resultados sobre las imágenes de entrada.

Para validar el sistema se han utilizados varios *datasets* con el fin de ponerlo tanto en situaciones reales de uso como en situaciones más extremas. Para este primer caso se ha grabado un *dataset* en el laboratorio de robótica del edificio I3A en la EINA y para el segundo se ha utilizado el *dataset* público *WildTrack*. Además, para validar los módulos por separado, se han utilizado varios *datasets* públicos para evaluar detección y re-identificación de personas.

Ya que es el paso clave del sistema, también se han buscado alternativas para ver cómo se podría mejorar el proceso de re-identificación. Una de estas mejoras ha sido utilizar mascaras de segmentación, cuyos resultados son beneficiosos solo bajo ciertas circunstancias. En general se ha observado que sería necesaria una segmentación más precisa y más tiempo de entrenamiento para alcanzar resultados más favorecedores. Lo que sí se ha visto esencial, como cabía esperar, para un buen funcionamiento, es el entrenamiento del descriptor de personas con datos de dominios similares al entorno de aplicación donde se quiere desplegar el sistema.

Entre los resultados obtenidos se ha podido observar cómo el sistema integra métodos del estado del arte en estos temas y se comporta correctamente, siendo capaz de predecir el aforo de un área determinada en situaciones de densidad de personas limitada, que era el objetivo, aunque cómo se podría esperar pierde precisión tanto en detección como en re-identificación en situaciones con más aglomeraciones de gente.

Índice general

Capítulo 1

Introducción

La detección y re-identificación de personas son dos campos de la visión por computador que han ido ganando cada vez más relevancia y se han visto aplicados en muchos aspectos cada día más comunes, desde la detección de peatones en vehículos autónomos hasta la re-identificación de personas en redes de vídeo-vigilancia. En concreto este trabajo se centra en una aplicación relacionada con este segundo ámbito, de redes de vídeo-vigilancia y el conocer el aforo de personas en un entorno determinado.

1.1. Motivación

La motivación inicial de este proyecto va ligada a una de las numerosas restricciones a las que la sociedad de hoy en día ha tenido que adaptarse debido a la pandemia COVID-19. Esta restricción no es otra que el aforo limitado en espacios públicos (tiendas, gimnasios, cines, etc.). Por lo tanto, el diseñar un sistema capaz de controlar cuánta gente hay en un instante de tiempo determinado en un área determinada ha pasado a ser un problema mucho mas común en nuestra vida cotidiana.

Adicionalmente, otro aspecto que ha motivado este trabajo ha sido el poder obtener conocimientos sobre una gran variedad de herramientas y métodos de detección y re-identificación de personas, dentro del mundo de la visión por computador y el aprendizaje automático, y cómo estas son utilizadas en otros proyectos para tratar diversas situaciones.

1.2. Descripción del problema

El problema a abordar en este trabajo corresponde al desarrollo de un sistema capaz de controlar el aforo de un entorno determinado mediante la información recopilada por varias cámaras. Este se puede descomponer en problemas más pequeños entre los que se encuentran: recopilación y puesta en común de la información (instantes de tiempo o *frames*) proporcionada por cada cámara, detección de personas en los diferentes instantes, re-identificación de las diferentes personas detectadas, escritura de resultados y evaluación. Todo ello sin olvidarse de los diferentes retos de integración de las numerosas herramientas utilizadas.

También es importante destacar que un sistema como este debería ser flexible a un número N de cámaras utilizadas, e independiente a la posición de estas en el entorno analizado. Esto significa que solapes de visión, obstrucciones parciales o totales y fallos en lecturas de imágenes no supongan un funcionamiento erróneo. Un ejemplo de entorno puede verse en la figura 1.1.

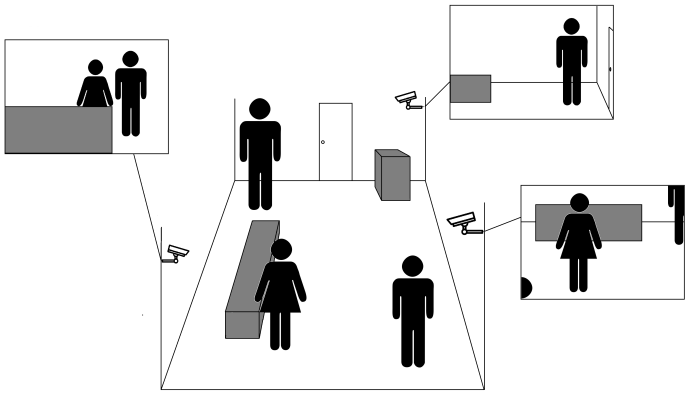


Figura 1.1: Sistema multi-cámara controlando un área determinada. Se puede apreciar cómo los diversos objetos obstruyen la visión de ciertas personas, y el campo de visión de las cámaras se solapa.

1.3. Estado del arte

A continuación se va a dar contexto a los componentes esenciales del sistema a desarrollar, la detección y re-identificación de personas de manera automatizada, así como ejemplos de aplicaciones similares existentes.

1.3.1. Detección de personas

El problema de detección de personas consiste en determinar la existencia de una persona en una posición e instante de tiempo determinados, como se muestra en la figura 1.2. Es importante diferenciar entre detecciones directas mediante dispositivos físicos (sensores) y detección indirecta mediante algoritmos que toman como entrada la información de sensores, ya que ambas tienen diferentes usos y campos de uso.

Algunos de los sensores que se pueden utilizar para detectar directamente la presencia de una persona son, por ejemplo: sensores ópticos, sensores infrarrojos, seguidores de wifi o sensores térmicos. Aunque resultan sencillas de instalar y utilizar, todas estas opciones tienen ciertas limitaciones, ya que dan información limitada, detectando la posición aproximada en el entorno o tan solo midiendo la cercanía relativa al sensor. Por lo tanto, cuando la tarea a realizar necesita un mayor nivel de detalle, estos sensores suelen ser utilizados en combinación a los algoritmos más avanzados.

Los algoritmos existentes más utilizados en los últimos años giran en torno a técnicas de Aprendizaje Automático, en particular de técnicas de *Deep Learning*

[1], y deben su uso a sus buenos resultados en los últimos años para numerosas tareas de reconocimiento visual automático. Estos se conocen como detectores, de personas en este caso, y se basan en el uso de redes neuronales entrenadas para reconocer instancias de personas en una imagen. Este entrenamiento, realizado con grandes bases de datos con muchos ejemplos, se traduce en aprender características de imagen muy distintivas de personas, que permiten al detector distinguir instancias de personas de cualquier otro elemento de la escena.

Existen muchas propuestas de detectores, entre los que presentan mejores resultados en la actualidad se encuentran YOLOv4 [2] o Detectron2 [3]. Ambos poseen cualidades destacables como la velocidad de YOLO, permitiéndole ser utilizado en detección en tiempo real, o la precisión de Detectron2 el cual proporciona qué área exacta ocupa la persona en la imagen. En las Figuras 1.2 y 1.3 vemos un ejemplo del tipo de detección que realiza cada uno de estos algoritmos. Ambos modelos se han considerado en este trabajo, y en el capítulo siguiente se describen en más detalle.

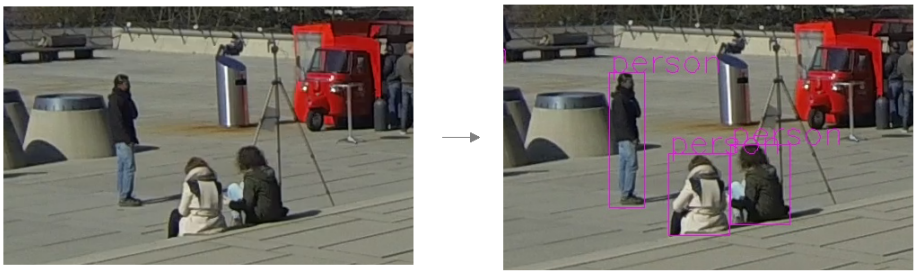


Figura 1.2: Ejemplo de detección: Salida del algoritmo de predicción YOLOv4 (fuente: imágenes extraídas del dataset *WildTrack* [4])

Figura 1.3: Ejemplo de detección: Salida del algoritmo de predicción Detectron2 sobre el dataset grabado en la EINA.

1.3.2. Re-identificación de personas

El hecho de utilizar cámaras para el análisis de personas, permite implementar tareas de más alto nivel que solo la detección, por ejemplo analizar cuándo la misma persona aparece otra vez en las imágenes. La re-identificación de una persona consiste en, dadas dos imágenes de instancias de personas, determinar si ambas corresponden a la misma persona. Esto se consigue mediante la extracción de características, también conocidas como descriptores los cuales

representan de manera numérica la imagen de la persona, para posteriormente poder comparar las características de ambas imágenes. Los modelos existentes están basados en técnicas de *Deep Learning* y siguen un entrenamiento y funcionamiento similar al de los detectores mencionados anteriormente.

Entre algunos modelos se pueden encontrar Osnet [5] [6], Multi-Level Factorisation Net (MLFN) [7] o Harmonious Attention Network (HACNN) [8].

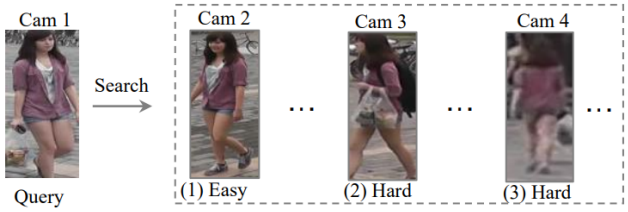


Figura 1.4: Ejemplo de varias vistas distintas de personas que deberían ser re-identificadas como iguales. El re-identificador no debe fallar pese a la diferencia entre las posiciones de la persona.(Fuente [9])

1.3.3. Conteo de personas

Una vez vistas herramientas de detección y re-identificación, se puede explicar cómo contar el total de personas en un área.

En casos donde se requiera un sistema de conteo genérico sencillo, sin necesidad de realizar re-identificación, por ejemplo, se podría hacer uso de los sensores de detección directa mencionados anteriormente junto con un software capaz de procesar los datos de dichos dispositivos. Un ejemplo de situación donde interese este tipo de conteo puede ser en un estadio donde se requiera un conteo aproximado de una alta densidad de personas.

En caso de necesitar re-identificación, para poder ejecutar un análisis de personas en entornos más complejos con varios puntos de entrada y salida o con frecuentes oclusiones, se requieren de sistemas mas cercanos al que se desea construir en este trabajo.

En la figura 1.5 podemos observar un caso donde solo interesaría un conteo aproximado con respecto a una situación donde sería interesante aplicar un conteo más preciso del aforo.

1.4. Objetivos y tareas

El principal objetivo es el desarrollo de un sistema capaz de determinar el aforo de un espacio monitorizado por varias cámaras con campos de visión que pueden estar superpuestos u obstruidos, estudiando diversos métodos de detección y re-identificación de personas. Los entornos para los que está pensado dicho sistema serán entornos cuya densidad de personas no sea alta, como en la sub-figura b de la figura 1.5.

Se han utilizado varios datasets de carácter público y, tras evaluar el sistema base con ellos, se han propuesto y evaluado ciertas modificaciones con el fin de refinar el funcionamiento del sistema en un entorno real con datos de ejemplo capturados en este proyecto.

(a) (b)

Figura 1.5: (a) Ejemplo de escenario donde se estima la densidad de personas aproximada . (b) Ejemplo de escenario donde resulta factible contar el numero exacto de personas.(Fuentes: https://www.flickr.com/photos/notarim/5586060515) y https://blogs.nottingham.ac.uk/makingsciencepublic/2012/06/21/science-communication/lab/)

En cuanto a la metodología seguida se han realizado las siguientes tareas, desglosadas en los diferentes puntos de interés dentro del trabajo:

- Estudio de bases teóricas y realización de tutoriales sobre *Deep Learning* con el fin de revisar conceptos básicos y aprender a utilizar las librerías más comunes en este tipo de técnicas.
- Búsqueda y recopilación de varios modelos de detección y re-identificación de personas y varios datasets para la experimentación.
- Diseño e implementación del sistema, incluyendo la integración de los diferentes modelos seleccionados, tanto de detección como re-identificación de personas.
- Diseño, implementación y evaluación de mejoras con el fin de aumentar el rendimiento del sistema.
- Toma y evaluación de datos propios tomados en las instalaciones de la EINA. Con el fin de conseguir una demo realista, ajena a datasets públicos.
- Documentación del código y redacción de la memoria.

En la figura 1.6 se ilustra mediante un diagrama de Gantt la distribución de las tareas en el tiempo asignado. En al tiempo que se ha dedicado al proyecto en conjunto se estima alrededor de las 430 horas, en la tabla 1.1 se muestra una aproximación del tiempo invertido en cada tarea.

1.5. Entorno de trabajo

Para el desarrollo del trabajo se han utilizados las siguientes herramientas:

- Lenguaje de programación python. Entre las diferentes librerías utilizadas encontramos *OpenCV* para facilitar el tratamiento de imágenes, y *tensorflow* [10] y *PyTorch* [11] para el uso de los diferentes modelos de detección y re-identificación basados en redes neuronales.

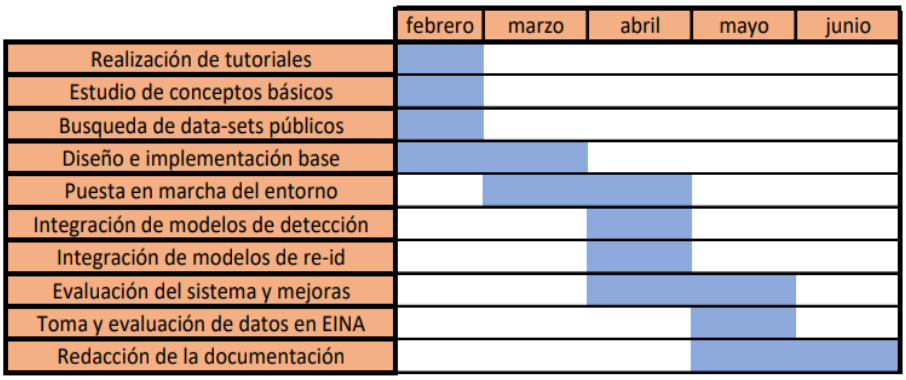


Figura 1.6: Diagrama de Gantt

Tarea	Tiempo(horas)
Realización de tutoriales	10
Estudio de conceptos básicos	15
Búsqueda de datasets públicos	10
Diseño e implementación base	80
Puesta en marcha del entorno	30
Integración de modelos de detección	60
Integración de modelos de re-id	50
Evaluación del sistema y mejoras	45
Toma y evaluación de datos en EINA	60
Redacción de la memoria	55
Total	415

Tabla 1.1: Tiempo aproximado dedicado a cada tarea propuesta del proyecto.

- Entre los diferentes detectores utilizados se encuentran YOLOv4 y Detectron2.
- Entre los diferentes modelos de re-identificación se han utilizado Osnet.

1.6. Resumen de la memoria

En el capítulo segundo se expondrá el trasfondo teórico relacionado con este trabajo así como una breve descripción de los algoritmos utilizados para detección y re-identificación, en capítulo tercero se hablará de la estructura del sistema y el *work flow* de este, y finalmente en el capítulo cuarto se expondrán los resultados obtenidos en los diferentes experimentos realizados.

Capítulo 2

Técnicas de detección y re-identificación

Tal y como se ha mencionado en la introducción, los métodos más prometedores hoy en día en tareas de detección y re-identificación de personas están basados en técnicas de *Deep Learning*. Este capítulo resume los conceptos más básicos del *Deep Learning* aplicado a reconocimiento visual en imágenes, y posteriormente se detallan las técnicas utilizadas en el sistema desarrollado para abordar los problemas de detección de personas y la re-identificación de estas.

2.1. Conceptos básicos de Deep Learning

El *Deep Learning* [12], como se ha adelantado en la introducción, es un área del aprendizaje automático que se basa en el uso de redes neuronales profundas. El objetivo de este tipo de redes es aprender representaciones discriminativas de los distintos elementos de interés a partir de numerosos datos de ejemplo, y a diferentes niveles de abstracción, con el fin de posteriormente ser capaces de realizar predicciones sobre nuevos datos nunca antes vistos por la red.

2.1.1. Elementos básicos y funcionamiento

Para ser capaz de retener la información de los datos de entrada, las diferentes representaciones de dichos datos son propagadas a lo largo de varias capas que componen la estructura de cualquier red neuronal. Las capas que se pueden encontrar en una red neuronal se pueden dividir en tres categorías:

- **Capa de entrada**: punto de entrada de los datos y primera capa que es visitada por estos.
- **Capas ocultas**: una o más capas encargadas de realizar el procesamiento de los datos recibidos por capas previas.
- **Capa de salida**: última capa de la red donde se reciben los resultados o predicción obtenidos del procesamiento de los datos de entrada.

Cada capa de una red neuronal está formada por unidades llamadas neuronas, cada una conectada con otras neuronas de capas previas y siguientes

formando entradas y salidas unas con otras. En la figura 2.1 se muestra un esquema de una red neuronal pequeña con dos capas ocultas. Las diferentes flechas entre neuronas representan las conexiones entre estas donde se aprecia el flujo de datos entrada/salida entre capas. Es importante destacar que la profundidad de la red viene determinada por el número de capas ocultas que existen entre la capa de entrada y la de salida, a mayor número de capas ocultas mayor es la profundidad, y más compleja la jerarquía de representación de los datos que puede aprender el modelo.

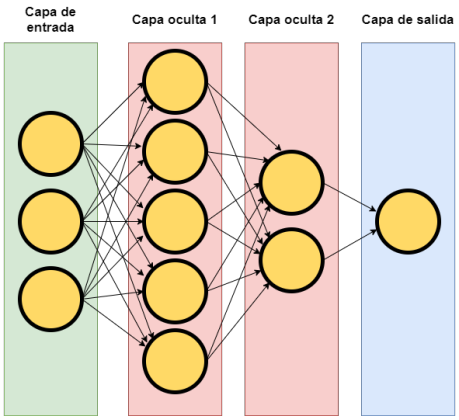


Figura 2.1: Ejemplo de una pequeña red neuronal con dos capas ocultas.

El último elemento básico que resta mencionar son los valores numéricos asociados a cada capa de la red, también conocidos como pesos. Inicialmente los valores de los pesos son aleatorios y el objetivo del entrenamiento de una red es modificar dichos pesos a unos valores con los que posteriormente obtener predicciones acertadas. Durante el proceso de entrenamiento, las diferentes neuronas reciben datos de entrada y estas modifican dichos datos según sus pesos, tras esto proceden a enviar a las siguientes neuronas los valores calculados propagando así la información a lo largo de la red. En caso de llegar a la capa de salida, en vez de propagar hacia delante los valores, se realiza la predicción final. Una vez realizada la predicción, durante el entrenamiento se ajustan los pesos de cada neurona dependiendo del error en la predicción realizada, que se evalúa con la función de pérdida establecida en dicha red.

Una vez finaliza el entrenamiento esto significa que los pesos de la red no se modificarán con nuevos datos de entrada, sino que se limitará a dar predicciones según estos.

En cada neurona también encontramos las funciones de activación, las cuales determinan según un umbral y una determinada operación la propagación de la información calculada. Entre las funciones de activación más comunes encontramos *ReLU*, *Softmax*, *tanH* y la *Sigmoid*. En la figura 2.2 se puede apreciar la estructura de una neurona y los diferentes elementos expuestos.

2.1.2. Redes Neuronales Convolucionales (CNN)

Un tipo de red neuronal profunda, y que es la más utilizada como base para tareas de reconocimiento en imágenes, son las Redes Neuronales Convolucio-

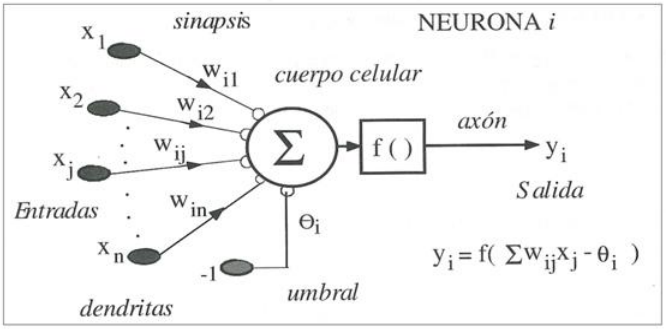


Figura 2.2: Diagrama de los distintos componentes de una neurona. (Fuente: http://grupo.us.es/gtocoma/pid/pid10/RedesNeuronales.htm)

nales [13] [14] (*Convolutional Neural Network* o CNN para abreviar). Dichas redes siguen utilizando capas para trasmitir la información a lo largo de la red pero añaden la idea de aplicar convoluciones, mediante filtros basados en conjuntos de *kernels* de una o más dimensiones. Los principales tipos de capas que encontramos en las CNN son:

- **Capa convolucional**: capa encargada de tomar conjuntos de píxeles cercanos y operarlo junto con matrices pequeñas llamadas *kernels*. El objetivo es, supongamos dado un *kernel* 3x3 (matriz 3x3), aplicar el producto escalar entre el *kernel* y conjuntos de píxeles de la imagen original para generar una nueva matriz resultante. En la figura 2.3 se muestra un ejemplo de convolución.

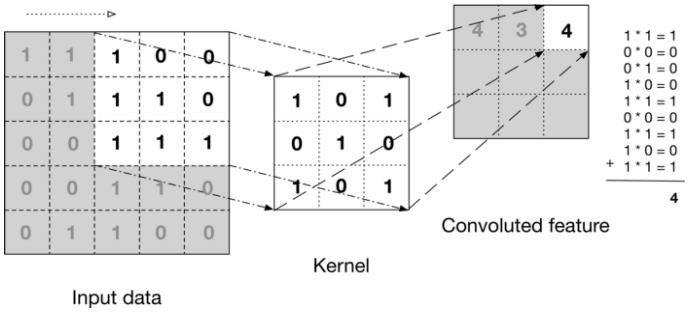


Figura 2.3: Ejemplo de convolución: *kernel* 3x3 aplicado a datos de entrada (*input data*). En la matriz de la derecha se ve el resultado (*convoluted feature*) que se obtiene en cada posición de la matriz al ir aplicando el *kernel* en las primeras posiciones de la matriz de entrada. (Fuente: https://medium.com/machine-learning-for-li/different-convolutional-layers-43dc146f4d0e)

- **Capa de activación**: capa cuya tarea es aplicar la función de activación a la salida de capas convolucionales.
- **Capa *Fully Connected***: capa final, similar a la capa de salida en una red neuronal básica. Realiza la clasificación determinada según los datos de entrada que reciba.

- **Capa de agrupación (*Pooling*)**: capa que ayuda a reducir el coste en tiempo computacional y adicionalmente ayuda a que gradualmente el modelo consiga ser invariante espacialmente, haciendo que se reduzcan el número total de parámetros y cálculos que realiza la red. En la figura 2.4 se muestra un ejemplo de dicha capa.

  Este tipo de capas suelen aparecer entre dos capas convolucionales para ayudar al modelo a retener y aprender las características más importantes entre capas. Suelen hacer uso de *kernels* para realizar el agrupamiento de características.

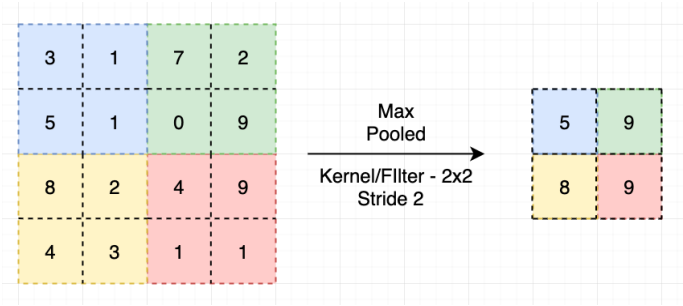


Figura 2.4: Ejemplo de capa de agrupación con *kernel* 2x2 donde se extrae el valor máximo de los píxeles seleccionados. (Fuente: https://ai.plainenglish.io/pooling-layer-beginner-to-intermediate-fa0dbdce80eb)

Finalmente en la figura 2.5 se muestra una arquitectura general de una CNN para visualizar las diferentes capas mencionadas en sus respectivas posiciones.

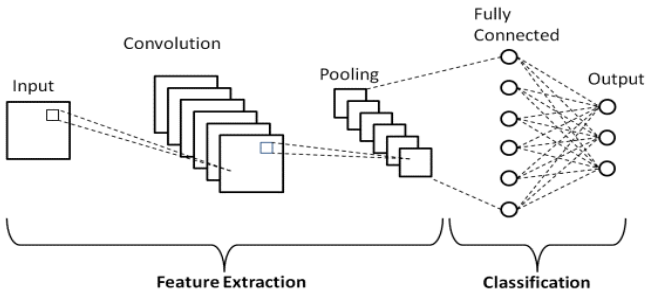


Figura 2.5: Ejemplo de estructura general de una CNN con una única capa convolucional, una capa de *pooling* y una capa *Fully Connected* . (Fuente: https://www.upgrad.com/blog/basic-cnn-architecture/)

2.2. Detección y re-identificación con CNNs

Una vez conocida la estructura y los elementos comunes que forma una CNN, a continuación se describe la aplicación de estas en la detección y re-identificación de personas, que es el problema a abordar en este trabajo, y de los modelos utilizados en este proyecto.

Los datos de entrada en un problema de re-identificación son imágenes tomadas a la misma persona, desde distintos sensores y en el mismo instante de tiempo. En ellas existirán numerosos factores que alteraran cada imagen: iluminación, resolución de imagen, ángulo de visión, deformaciones en la lente, oclusiones en el campo de visión, etc. Y aquellos modelos CNN utilizados deberán ser suficientemente robustos como para utilizar distintos tipos de imágenes de manera exitosa, tanto para detección como re-identificación.

Como retos principales en cada uno de los dos modelos necesarios, cabe destacar:

- En la **detección** de personas: detectar a la persona aunque parte de la misma no sea fácilmente identificable.
- En la **re-identificación** de personas: emparejar correctamente dos vistas distintas de la misma persona pese a variaciones entre las dos imágenes en cuanto a iluminación, posición u oclusión.

2.2.1. Modelos CNN utilizados en detección

Para este trabajo se han seleccionado como modelos para detección de personas los detectores **YOLOv4** [2] y **Detectron2** [3]. Ambos métodos están diseñados para la detección más general, no solo personas, con la diferencia del nivel de detalle en las detecciones: YOLO proporciona cajas de aquellos elementos detectados, mientras que Detectron2 proporciona cajas además de una segmentación detallada de los contornos. Como se puede ver en las figuras 1.2 y 1.3 proporcionadas en el capítulo introductorio.

Los dos detectores están basados en CNNs y son ampliamente utilizados debido a su buen rendimiento, como se detalla a continuación:

- ***YOLOv4***: modelo construido sobre su predecesor *YOLOv3* [15] y basado en la misma CNN para la extracción de características, *CSPDarknet53* [16]. Dicha red cuenta con 53 capas y está diseñada para reducir el coste en tiempo computacional al mismo tiempo que mejorar los cálculos del gradiente. Todo esto conseguido gracias al particionado de características de la capa base en dos parte que posteriormente son unidas mediante una jerarquía. Dicha idea se repite en varias capas para conseguir la velocidad que caracteriza a este modelo. Sus aspectos más determinantes para ser utilizado en este proyecto han sido la velocidad y precisión de clasificación de imágenes, permitiendo detectar con precisión instancias de personas pese a oclusión de las mismas. La Figura 2.6 resume la estructura de la CNN Darknet para poder observar la organización de sus capas.
- ***Detectron2***: Este modelo se construye sobre *ResNet* [17], un tipo de CNN que trata de resolver el problema de desvanecimiento de gradiente. Este problema consiste en que el gradiente alcanza valores muy pequeños que impiden que los pesos de las capas modifiquen sus valores. Dicho problema se resuelve mediante ciertas propagaciones de los valores del gradiente hacia capas más profundas. En la figura 2.7 se muestra la estructura de la red ResNet y la organización de sus diferentes capas.

  Una característica que ha hecho interesante el uso de Detectron2 en el proyecto han sido su precisión a la hora de detectar personas en diferentes posturas. Adicionalmente, el poder obtener el área que dicha persona

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Table 1. **Darknet-53.**

Figura 2.6: Estructura de la CNN Darknet-53. Fuente [15]

detectada está ocupando se ha utilizado también para realizar ciertas pruebas en las que se elimina el fondo y se mantiene tan solo la propia persona detectada.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Figura 2.7: Estructura de la CNN ResNet50. (Fuente: `https://iq.opengenus.org/resnet50-architecture/`)

2.2.2. Modelos CNN utilizados en re-identificación

En la literatura encontramos numerosas soluciones para el problema de re-identificación de personas en las que CNNs son una buena solución [9].

En particular, como se ha comentado, para este trabajo se ha seleccionado el modelo *OsNet* [6] para tratar dicho problema. El uso de este modelo se barajó inicialmente debido a que es una CNN relativamente nueva, proporciona facilidad de uso y trata de abordar el problema de la re-identificación como objetivo

principal.

La arquitectura *OsNet* se basa en el aprendizaje de características por medio de funciones *omni-escalares*. Esto se consigue mediante la separación de las capas convolucionales tradicionales en varios flujos convolucionales que tratan de detectar las diferentes características de los datos de entrada en escalas distintas, un ejemplo de esto se puede ver en la figura 2.8. Esas separaciones se hacen en función a un *kernel* determinado de antemano. Con esta estrategia se reduce significativamente el número de parámetros y ayuda a ajustar de mejor manera los pesos de las características dependiendo de su escala.

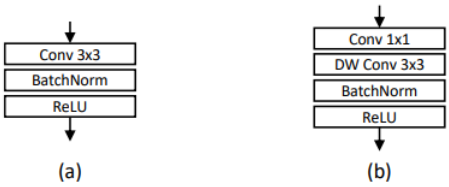


Figura 2.8: Ejemplo de cómo OsNet varía las capas convolucionales con lo que denominan *Lite*. (a) Standard 3×3 convolution. (b) Lite 3×3 convolution. DW: Depth-Wise. Fuente [6]

La estructura del modelo *OsNet* se muestra a continuación en la figura 2.9.

La característica principal por la que se ha decidido utilizar *OsNet* ha sido su velocidad, siendo una red ligera capaz de procesar rápidamente numerosos datos de entrada. Adicionalmente la misma ligereza que le proporciona velocidad en procesado también la hace sencilla de entrenar, facilitando entrenamientos que posteriormente se han planteado para mejorar la re-identificación.

stage	output	OSNet
conv1	128×64, 64 64×32, 64	7×7 conv, stride 2 3×3 max pool, stride 2
conv2	64×32, 256	bottleneck × 2
transition	64×32, 256 32×16, 256	1×1 conv 2×2 average pool, stride 2
conv3	32×16, 384	bottleneck × 2
transition	32×16, 384 16×8, 384	1×1 conv 2×2 average pool, stride 2
conv4	16×8, 512	bottleneck × 2
conv5	16×8, 512	1×1 conv
gap	1×1, 512	global average pool
fc	1×1, 512	fc
# params		2.2M
Mult-Adds		978.9M

Figura 2.9: Estructura de la red OsNet con dato de entrada de tamaño 256 × 128.

Capítulo 3

Sistema de control de aforos

El sistema desarrollado para control de aforos automático, basado en sensores de visión, cuenta con tres módulos detallados a continuación. El sistema se ha desarrollado con una arquitectura modular que permite intercambiar versiones o variaciones de dichos módulos sin interferir con el diseño, siempre y cuando se respeten los formatos de entrada y salida de cada módulo.

3.1. Estructura del sistema

De manera general, la entrada del sistema son las diferentes imágenes de las N cámaras y la salida de este son las mismas imágenes con los resultados del conteo de personas en la escena visualizados (superpuestos) sobre ellas. A continuación se muestra un diagrama del sistema en la figura 3.1 y los diferentes pasos que las imágenes de entrada deben seguir para ser procesadas y poder calcular el aforo contenido entre ellas.

Antes de detallar los módulos del sistema, se describen dos aspectos importantes relativos al diseño del sistema: cómo compartir la información de las cámaras y algunas suposiciones sobre cómo estas visualizan el área a monitorizar.

Puesta en común de la información de las cámaras. Las diferentes cámaras que son la fuente de información para el sistema proporcionarán imágenes RGB para cada instante de tiempo.

Al *frame* capturado en cierto instante de tiempo t, desde la cámara número i, lo denominamos $frame_i(t)$. Por lo tanto, para cada instante de tiempo que se desee analizar se extraerá el *frame* capturado por cada cámara para dicho instante. La información de las cámaras se procesa (se comparte, recibe y gestiona) completamente por el sistema de manera **centralizada y secuencial**.

Se ha elegido esto frente a un sistema distribuido debido a que, pese a que es más robusto, una implementación distribuida es más compleja y es mejor opción cuando existe riesgo de fallos y caídas en el sistema. Por otro lado, una versión centralizada permite evitar todos los desafíos que genera un sistema distribuido a costa de menos seguridad en aspectos de caída o fallos de la máquina donde se encuentre.

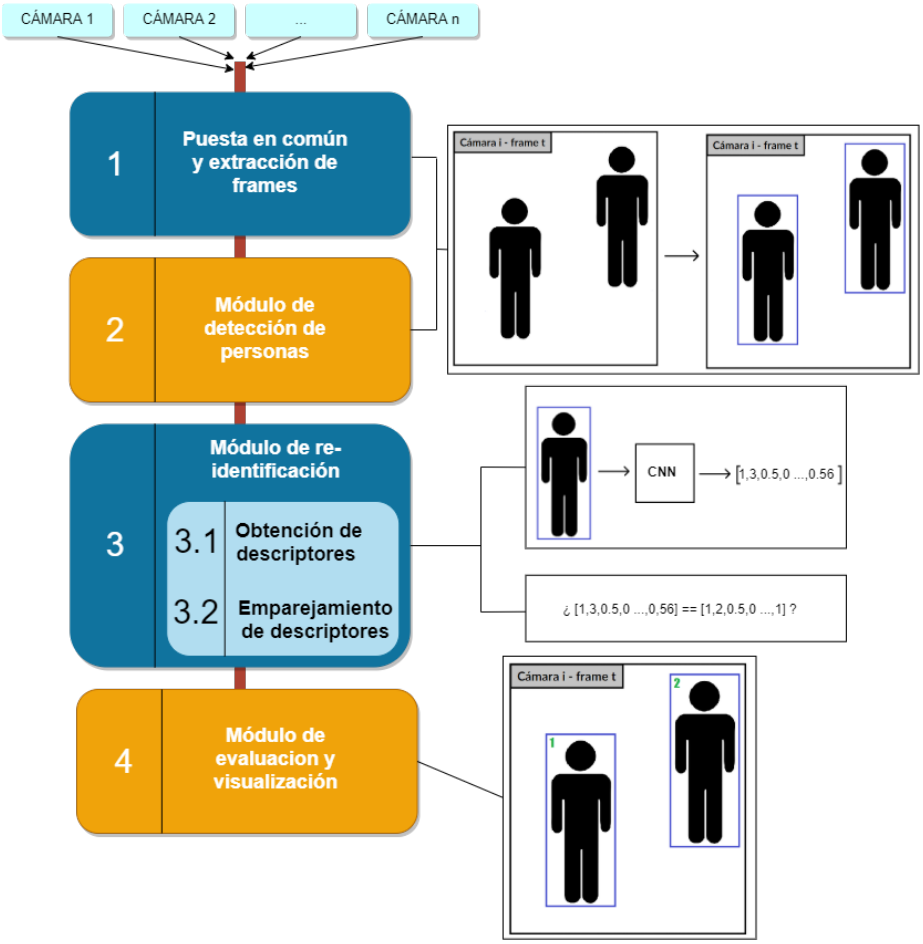


Figura 3.1: Diagrama de las etapas principales del sistema.

Otro aspecto de interés es el hecho de que, para flexibilizar el uso del sistema, este puede procesar tanto vídeos o *frames* extraídos directamente de estos.

Suposiciones realizadas. Las principales suposiciones realizadas al comienzo y a lo largo del desarrollo del trabajo que afectan a cómo se enfoca el problema y las restricciones que se dan a este son:

- El número de cámaras será indefinido. Esto quiere decir que no es un número fijo y puede ser variable con el tiempo. En el caso de los experimentos futuros se ha probado desde 4 a 7 cámaras.
- La posición de las cámaras, y posibles solapes u oclusiones en el campo de visión de las mismas no debe ser crítico para el correcto funcionamiento del sistema.
- La sincronización de las cámaras se supone correcta en todo momento. Esto quiere decir que todos los *frames* proporcionados en un instante de

tiempo ($frame_1(t)$ ... $frame_N(t)$) deben pertenecer al mismo instante de tiempo en todas las cámaras.

Estas suposiciones facilitan la abstracción entre el sistema y el entorno monitorizado para evitar dependencias con la posición, visión o rango de vistas disponibles del área.

3.2. Módulo de detección

Este primer módulo (módulo 2 en la figura 3.1) recibe como entrada las imágenes capturadas por las cámaras en un instante de tiempo y da como salida las personas detectadas en las diferentes imágenes.

Una vez recibidos los *frames* del instante de tiempo procesado, este primer paso realiza la detección de instancias de personas en cada uno de dichos *frames*. Para ello se ha desarrollado un módulo que aplica dicha detección dado un modelo existente seleccionado.

El proceso de detección consiste en determinar, para cada persona en el *frame* i-ésimo, su *bounding box* mínima. Esto quiere decir, obtener la caja que encierra a la persona detectada con un área mínima, como se aprecia en la figura 3.2

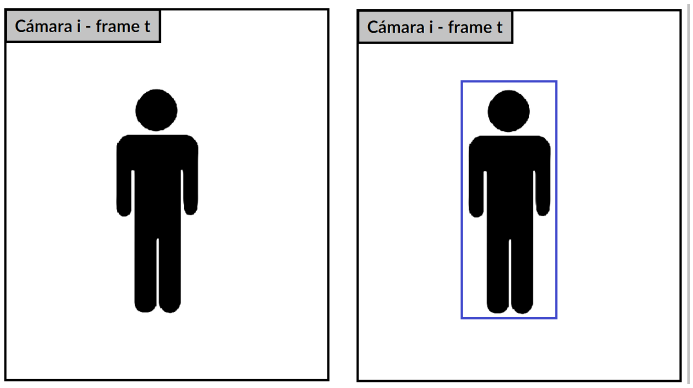


Figura 3.2: Ejemplo de *bounding box* que recubre la persona detectada.

Como se ve en más detalle en el diagrama de la arquitectura del anexo A.1, para facilitar el cambio entre distintos detectores, se ha implementado un *abstract_detector*, que solo tiene dos métodos en los cuales se encapsula su funcionamiento: un método *init* para la creación de una instancia de dicho detector y un método *predict* para realizar la predicción sobre una imagen. Este predictor abstracto es la clase en la que se incluyen los detectores utilizados, que como se ha mencionado en el capítulo anterior, en particular han sido YOLOv4 y Detectron2 (se hace uso de un parámetro de configuración para identificar qué predictor utilizar en el arranque del sistema, revisar anexo A.2 para más información sobre el sistema y sus parámetros).

3.3. Módulo de re-identificación

Este segundo módulo (módulo 3 en la figura 3.1) recibe como entrada las detecciones realizadas sobre las personas en las imágenes y realiza el proceso de

re-identificación entre ellas.

Una vez obtenidas las personas detectadas en cada *frame*, se deben pre-procesar para generar la entrada más precisa necesaria para el paso de re-identificación. En particular, es necesario recortar los fragmentos de la imagen indicados por las *bounding boxes* para pasar solo la información esencial al apartado de re-identificación como se muestra en la figura 3.3.

Los modelos de re-identificación de personas extraen los descriptores de cada una de estas imágenes (recortadas) proporcionadas; es decir, nos proporcionan un descriptor de cada persona que se ha detectado en el modulo anterior.

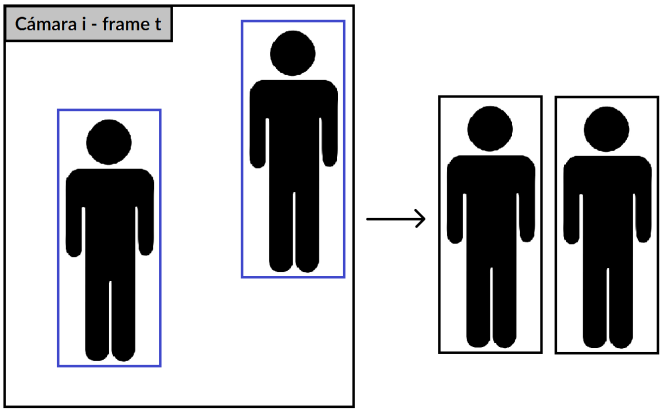


Figura 3.3: Ejemplo de recorte de las instancias de personas detectadas en una imagen.

Una vez se extraen todas las imágenes recortadas de las personas detectadas, se extrae para cada una un descriptor asociado. De manera análoga a como se gestiona la predicción en el modulo anterior (ver diagrama en el anexo A.1) se ha creado una clase *abstract_REID* encargada de encapsular el código de los diferentes modelos utilizados para la extracción de descriptores. Posee un método de inicio *init* y un método *predict* encargado de, dadas una serie de imágenes, obtener los descriptores de las mismas.

Tras obtener los descriptores ya se tienen todas las herramientas para realizar la re-identificación, para la cual se aplica el siguiente algoritmo:

- Cada persona detectada comienza con un identificador nulo.
- Se procede a analizar las personas de la cámara 0 hasta la N-ésima.
- Si se detecta una persona con identificador nulo, nos encontramos ante una persona nunca antes vista en el instante de tiempo por lo que le asignamos una nueva identificación válida. Tras esto se procede a comparar, en el resto de cámaras si algún descriptor coincide con el de la nueva persona detectada.

  Para determinar si dos descriptores son suficientemente iguales se ha hecho uso de la similitud del coseno, ec. (3.1). Dicha medida evalúa el coseno entre los dos vectores, siendo idénticos en caso de obtener un resultado de $cos(A, B) = 0$ o completamente ortogonales en caso de $cos(A, B) = 1$.

$$similarity = \cos\theta = \frac{A \cdot B}{\|A\|\|B\|} \tag{3.1}$$

Si la persona con la que se está comparando tuviera una identificación asignada previamente, sería necesario comprobar si el nuevo cálculo del coseno obtenido es mayor al que se obtuvo cuando se calculó con la anterior identificación.

- Para poder asignar una identificación mediante comparación, la similitud del coseno obtenida deberá estar dentro de un umbral determinado.

De esta manera se garantiza que todas las personas detectadas obtendrán una identificación, y perduraran aquellas re-identificaciones con mejor similitud de coseno. Finalmente se tendrá para cada *bouding box* un identificador asociado obtenido de sus descriptores.

3.3.1. Descripción más detallada de personas.

Tras las primeras pruebas del sistema base se estudiaron varias alternativas para intentar mejorar la re-identificación, ya que es el paso clave para poder realizar el conteo exacto de personas. Entre estas alternativas encontramos: la eliminación del fondo de las imágenes para segmentar a las personas y la de añadir ciertos meta-datos a las detecciones realizadas para poder utilizarlos en el apartado de re-identificación (posición de la persona, postura de la misma, etc.).

Eliminación de *background*. Se implementó la opción de obtener descriptores más precisos de las personas reduciendo el ruido introducido por el fondo de las imágenes. En particular, se utiliza un método de segmentación de imágenes; es decir, separar la persona detectada del fondo siguiendo el contorno de dicha persona en la imagen. De esta manera todo lo que no sea parte de la persona quedaría categorizado como fondo y se eliminaría.

En la figura 3.4 se puede apreciar un ejemplo de persona detectada que ha sido segmentada para la extracción de su descriptor. En este ejemplo vemos el tipo de ruido que podemos evitar segmentando a la persona. En el fondo podemos observar un coche rojo e incluso el pie de una persona diferente. Estos elementos pueden influenciar a la hora de comparar descriptores, ya que, pese a que la persona no lleva nada rojo, en su descriptor se indicará la existencia de píxeles rojos en la posición del pecho, generando así la posibilidad de que posteriormente se realice una re-identificación errónea. También se muestra a su izquierda el resultado de eliminar el fondo de la imagen para mantener solo la persona reconocida.

Para conseguir esta eliminación del fondo de la imagen es necesario una máscara que indique el área que ocupa una persona en la imagen. Algunos detectores, como es el caso de Detectron2, facilitan dichas mascaras al mismo tiempo que las *bounding boxes*. Una vez se tiene la máscara se puede aplicar una operación AND bit a bit entre la imagen y la mascara para obtener la persona sin el fondo.

Por lo tanto, en caso de desear esta alternativa, hace falta un paso extra en la figura 3.1 como parte del pre-procesado antes de ejecutar el punto 3.

Figura 3.4: Ejemplo de imagen recortada tras la detección de una persona (izquierda) y la misma imagen aplicando una segmentación más exacta de la persona (derecha). Fuente https://paperswithcode.com/dataset/msmt17

Seria necesario realizar este proceso de eliminación de *background* a las personas detectadas.

Re-entreno de modelo de descripción (OsNet). Además, el trabajar con las personas segmentadas en más detalle (sin el *background*), también implica el re-entreno del modelo utilizado para la re-identificación. OsNet esta disponible entrenado con datos de varios datasets de carácter público entre los que se encuentran Market-1501, DukeMTMC, CUHK03 y MSMT17. El mismo modelo de OsNet se ha entrenado y evaluado con varias combinaciones de dichos datasets dando así una gran variedad de resultados. En los experimentos lo re-entrenaremos con colecciones de datos, de los mismos datasets mencionados, cuyo fondo ha sido eliminado. De esta manera el modelo puede aprender solo de las personas y se evitaría que el fondo añadiera demasiado ruido a lo largo del entrenamiento.

Para poder realizar dicho re-entrenamiento los pasos a seguir han sido:

- Se recopilaron los diferentes datasets utilizados en el entreno original del modelo de OsNet_ain_x1, último modelo de OsNet el cual obtiene mejores resultados comparados con versiones anteriores [1] (en el sistema se utiliza este modelo entrenado con MSMT17 y evaluado con Market1501 y DukeMTMC). Entre los datasets que se han utilizado están MSMT17 [18] y MARKET_1501 [19].

- Se utilizó Detectron2 para obtener las máscaras de segmentación de las personas detectadas y poder procesar las imágenes de acuerdo al método expuesto en el apartado de eliminación de *background*. De esta manera se ha replicado cada dataset con las personas detectadas en la versión original y con las personas detectadas eliminando el fondo.

- Se entrenaron nuevos modelos de descripción de personas (siguiendo la arquitectura de OSNET) con dos estrategias:

[1] https://kaiyangzhou.github.io/deep-person-reid/MODEL_ZOO

- Entrenamiento desde cero: Entrenamiento iniciado con pesos aleatorios en el que el modelo no posee nada aprendido.
- Fine-tuning: Consiste en una estrategia común para realizar *transfer learning* que es utilizada cuando no existen grandes cantidades de datos para entrenar un modelo, y además se desea aprovechar lo aprendido previamente por un modelo cuyo dominio sea similar al del objetivo a aprender. En este caso nuestro dominio (personas segmentadas) es similar al del modelo original (personas sin segmentar) por lo que se puede aplicar a los modelos utilizados. Este entrenamiento se realiza a partir de los pesos ya entrenados del modelo original por lo que es suficiente con cargarlos antes de comenzar el entrenamiento. Más rápido debido a que el modelo no comienza con pesos aleatorios sino con pesos que han convergido hacia resultados positivos.

Para realizar los entrenamientos se ha implementado un *script* donde se seleccionan los diferentes parámetros deseados: dataset de entrenamiento, dataset de evaluación, ratio de aprendizaje, épocas de aprendizaje, método de evaluación, etc.

3.4. Módulo de evaluación y visualización

El último módulo del sistema a comentar es el módulo utilizado en la evaluación del sistema, el cual no es usado siempre ya que requiere de tener el *ground truth* (anotaciones indicadoras de qué se debe reconocer y a qué clase pertenecen) de los datos que están siendo procesados. Por lo tanto, es un módulo utilizado en este trabajo sobre algunos datasets de los cuales se poseían las anotaciones de las personas en cada *frame*.

En este módulo se incluye la evaluación del funcionamiento de los dos módulos principales:

Evaluación de detección . Para evaluar la calidad de la detección de personas se utilizan las métricas típicas para estos problemas,

precision (P) y *recall* (R), ec.(3.2), para cada conjunto de frames que representan un instante determinado.

$$P = \frac{true\ positives}{true\ positives + false\ positives} \qquad (3.2)$$
$$R = \frac{true\ positives}{true\ positives + false\ negatives}$$

donde:

- Una persona detectada por el sistema y que se encuentra anotada en el *ground truth* es un *true positive*,
- una persona detectada por el sistema y de la cual no se encuentra una anotacion en el *ground truth* es un *false positive*,
- una persona no detectada por el sistema y que se encuentra anotada en el *ground truth* es un *false negative*,

- una persona no detectada y no se encuentra en anotaciones es un *true negative*.

Para determinar si la predicción realizada es correcta con respecto a las anotaciones se ha utilizado el cálculo de la intersección sobre la unión (o como son sus siglas en inglés, IOU). El IOU determina el área compartida por dos rectángulos y nos permite, en este caso, conocer si dos *bounding boxes* se superponen lo suficiente como para estar identificando la misma persona. Un ejemplo de uso se muestra en la figura 3.5 donde aparece la caja del *ground truth* con respecto a la caja predicha.

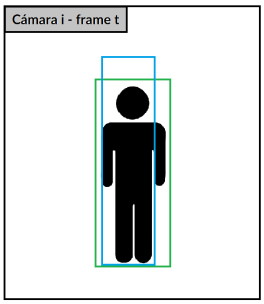


Figura 3.5: Ejemplo de diferencia en tamaño entre predicción (azul) y *ground truth* (verde) donde el cálculo del IOU obtendría un 60 % de área compartida.

Evaluación de re-identificación. En este paso se quiere determinar si la re-identificación de cada persona detectada en un *frames* ha sido correcta en el resto de *frames*. Lo que quiere decir, por ejemplo, que la persona p_i que aparezca en los *frames* 1 y 2 deberá tener el mismo identificador en ambas instancias.

Esto se realiza mediante la comparación de identificadores asignados entre pares de instancias de una persona, para determinar cuántas veces se ha realizado de manera correcta o errónea una re-identificación. Entre las métricas definidas tenemos:

- *pares_totales*: número completo de pares que han sido analizados por el sistema.
- *pares_ok*: número de pares que se han re-identificado correctamente por el sistema.
- *personas_mal*: número de personas que han tenido, al menos, un par mal re-identificado.

En la figura 3.1 se puede observar un ejemplo de esta evaluación, suponiendo un sistema que cuenta con 3 cámaras y solo 1 persona que identificar. La persona con identificador 105 en las anotaciones ha sido detectada principalmente con identificador 3 en el sistema (seleccionando el número más repetido y tomando la primera cámara con dicho identificador como referencia). Por lo tanto, tenemos 2 pares que analizar: cámara 1 con cámara 3 y cámara 1 con cámara 2. Obteniendo así un par incorrecto y otro correcto.

	Cámara 1	Cámara 2	Cámara 3
id reID	3	3	2
id anotaciones	105	105	105

Pares considerados: c1-c2 y c1-c3
Total de pares analizados (*pares_totales*): 2
Número de pares correctos (*pares_ok*): 1

Tabla 3.1: Ejemplo de evaluación de re-identificación por el sistema. Suponiendo un sistema con 3 cámaras y una sola persona a detectar. Se tienen en cuenta 2 pares ya que se toma la cámara 1 como referencia.

Este ejemplo se puede entender mejor si se ve una representación real del mismo. En la figura 3.6 podemos observar un caso donde una misma persona se identifica en un caso erróneamente. Se tomará el identificador 1 como principal, ya que es el más repetido, se evaluarán 2 pares, y se obtendrá que 1 de ellos es correcto.

Figura 3.6: Ejemplo de situación en la que se re-identifica erróneamente una persona en una cámara, pero en las otras 2 del sistema se re-identifica correctamente.

Capítulo 4

Experimentos realizados

En este último capítulo se exponen los diferentes experimentos realizados para evaluar cada componente del sistema implementado así como el comportamiento del sistema de manera global.

4.1. Entorno y datos de experimentación

4.1.1. Entorno de evaluación.

Para la realización de experimentos se ha trabajado en un ordenador equipado con un procesador i7 de 9ª generación, 32 GB de memoria RAM, una tarjeta gráfica RTX 2070 y sistema operativo Windows 10. Como se ha comentado, todo el trabajo está implementado en python y basado, sobre todo, en librerías de *opencv* y de *pytorch*. Para evaluar los diferentes experimentos para re-identificación y entrenamientos se ha utilizado el entorno de *torchreid* [20], creado por los autores de OsNet.

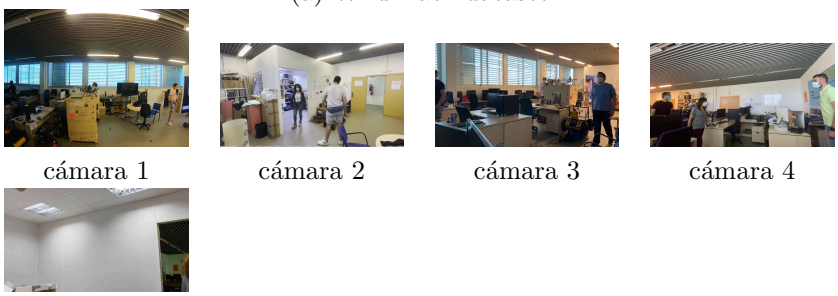
4.1.2. Datos utilizados.

Es importante destacar los distintos datasets con los cuales se han realizado pruebas. Según la naturaleza de estos datos, sobre todo en cuanto al tipo de etiquetado disponible, se ha diferenciado entre medidas cuantitativas (para observar, ajustar y validar el funcionamiento de los módulos) y otras más cualitativas.

WildTrack. Dataset que cuenta con 7 cámaras distintas monitorizando un espacio exterior correspondiente a una plaza frente a un edificio. Las personas que aparecen en escena son transeúntes que cruzan dicho espacio.

Para aspectos cuantitativos en los módulos de detección y re-identificación se ha hecho uso de este dataset *WildTrack*, ya que representa un entrono de cierta dificultad para ambos problemas debido a la alta densidad de personas. Un ejemplo de este dataset se muestra en la figura 4.1, donde se pueden apreciar conjuntos grandes de personas en posiciones diversas y con niveles altos de oclusión entre ellas.

(a) WildTrack dataset



(b) EINA dataset

Figura 4.1: Ejemplo de imágenes de los dos datasets principales utilizados. En cada bloque se muestra una imagen desde cada cámara del dataset correspondiente en un instante de tiempo determinado.

EINA dataset. Este dataset se grabó en el laboratorio de robótica del i3A en la EINA. Se compone de 5 vídeos, de 5 minutos, grabados por 5 cámaras (3 iPhones, 1 móvil Android y 1 GoPro) de manera simultanea. Adicionalmente, existen cambios de iluminación entre diferentes zonas del laboratorio monitorizado y diferentes ángulos de visión proporcionados por las cámaras. Gracias a esta variedad de elementos se asegura que el sistema es flexible a distintos tipos de sensores ópticos. Las personas que aparecen ocupan dicho laboratorio sin llegar a salir de él; por lo tanto, se sabe en todo momento cuánta gente debe haber dentro.

Este dataset se ha utilizado de manera cualitativa para poder observar el funcionamiento del sistema de manera general, pero a su vez también para evaluar cierto aspecto cuantitativo principal como es el conteo de personas.

En la figura 4.1 se puede observar un ejemplo de este dataset y el entorno monitorizado.

4.2. Validación del módulo de detección

Para comenzar con la validación es importante destacar que para ambos detectores utilizados se ha empleado la configuración por defecto, ya que no ha sido necesario realizar ningún cambio sobre esta. Para ambos se han cargado los pesos por defecto que se aplican con su instalación.

Como se ha explicado en el anterior capítulo, en el módulo de evaluación

se han utilizado tres métricas principales para evaluar el funcionamiento de la detección de personas: el calculo del IOU y la obtención de *precision* y *recall*.

Primero se obtienen las detecciones de personas de un instante de tiempo determinado. Conociendo aquellas que proporcionan las anotaciones de referencia de dicho instante, se calcula para cada persona de las anotaciones, la persona predicha con la que se alcance mayor IOU. Una vez seleccionada una persona detectada de las predicciones, si el IOU entre ellas supera cierto umbral, se entiende que se ha realizado una predicción correctamente; es decir, se cuenta como *true positive*.

Aquellas cajas predichas que no se han emparejado serán *false positives* y aquellas anotaciones para las que no se ha encontrado una predicción correcta serán *false negatives*.

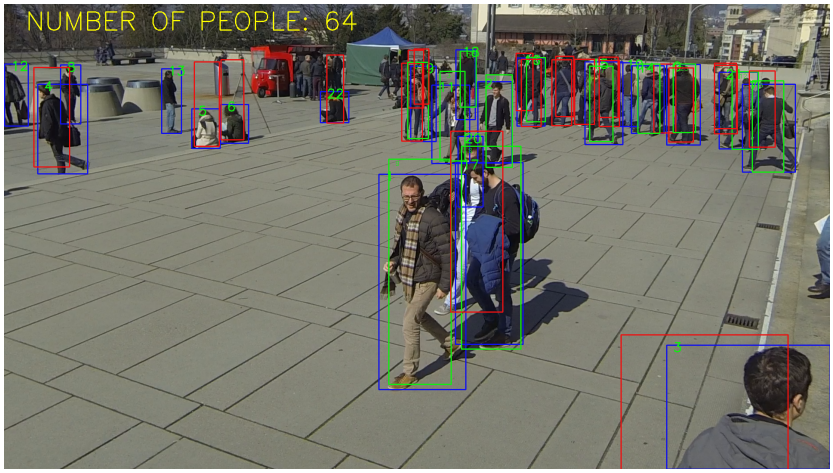
Es necesario utilizar el ratio de IOU para ajustar un cálculo realista de la *precision* y *recall* ya que dicho parámetro es el umbral que determina si una predicción se empareja con una anotación correctamente. Un IOU muy bajo suponía que cajas entre personas cercanas se confundieran entre ellas y un IOU muy alto generaba problemas, ya que las predicciones no son exactas a las anotaciones. Es importante destacar que debido a que las anotaciones del dataset *WildTrack* son mediocres, muchas personas distantes a la plaza principal no están etiquetadas, resultando en detecciones correctas que se categorizan como *false positives*.

En la figura 4.2 se pueden observar algunas de estas malas situaciones derivadas de malas anotaciones. En la subfigura **(a)** personas sentadas se detectan correctamente pero las anotaciones indican una caja como si estas estuvieran de pie. Adicionalmente se ve una gran cantidad de gente muy densa haciendo difícil su detección y puede darse el caso de que las cajas se confundan entre ellas debido a un bajo IOU. Por otro lado, en la subfigura **(b)** se puede apreciar un alto número de personas fuera de la plaza (escaleras) que son correctamente detectadas pero no aparecen en las anotaciones. También se pueden apreciar varias personas solapadas cuyas anotaciones son muy pobres e incluso erróneas, marcando cajas que no indican una persona en el *frame* o no cubriendo a la persona por completo.

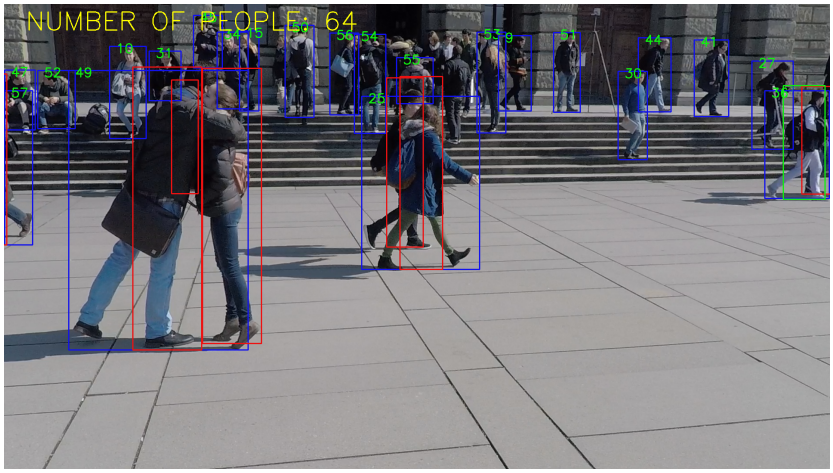
En la tabla 4.1 se muestra la *precision* y *recall* obtenida con ambos detectores con respecto al umbral de IOU asignado en la inicialización del sistema. El solape total entre las predicciones y anotaciones es imposible y, para tratar de ser realista, se ha variado el IOU desde el 0.5 (valor comúnmente utilizado en trabajos similares en la literatura) 0.1 hacia arriba y hacia abajo.

Se puede observar que el predictor YOLOv4 muestra un funcionamiento mejor y que, a medida que se vuelve más permisivo el umbral, más se incrementa el *recall*, ya que las personas predichas requieren de menos espacio compartido con las anotaciones.

Como era de esperar cuanto más permisivo se es con el umbral de IOU más detecciones se realizan. Sin embargo, para evitar detecciones muy ruidosas y acercarnos a lo común en trabajos similares, se ha decidido utilizar como valor predeterminado para el IOU el 0.5.



(a)



(b)

Figura 4.2: Ejemplo de los resultados del sistema en imágenes del dataset *WildTrack* (cámara 4). En azul las predicciones del sistema, en verde las cajas acertadas de las anotaciones y en rojo las cajas no acertadas de las anotaciones.

Detector	threshold	precision	recall
YOLOv4	0.4	0.42	0.61
	0.5	0.37	0.54
	0.6	0.27	0.4
Detectron2	0.4	0.34	0.68
	0.5	0.31	0.61
	0.6	0.23	0.47

Tabla 4.1: Tabla de resultados, *precision* y *recall* de detección de personas, variando el umbral del IOU. Las pruebas están realizadas en un conjunto de 40 *frames*

4.3. Validación del módulo de re-identificación

La evaluación de la re-identificación ha sido el punto principal y donde mayor número de experimentos se ha realizado, teniendo en cuenta que para el objetivo del sistema (contar cuántas personas aparecen en total) es imprescindible re-identificar lo mejor posible a las personas detectadas en distintas cámaras, para no contarlas más de una vez.

Como se ha mostrado en el capitulo anterior, la evaluación de la re-identificación se ha realizado mediante la comparación de pares entre las instancias de una persona detectada y re-identificada.

4.3.1. Resultados con modelos existentes de OsNet

Este experimento se ha realizado con la configuración por defecto para ambos detectores mostrada en el apartado de validación de detección. Todos estos experimentos utilizan el mismo modelo de OsNet para la descripción de las personas (en particular, el disponible *osnet_ain_x1_0* que estaba entrenado con MSMT17 y evaluado sobre Market1501[1]).

La aplicación o no de las máscaras proporcionadas por Detectron2 se decide internamente en el sistema mediante un parámetro de control.

La tabla 4.2 muestra cómo varían los resultados para estas opciones de detección con respecto al umbral de la similitud del coseno utilizado.

Resultados sin máscara de segmentación en detección. La re-identificación parte de la detección de cada detector y en este caso se aprecia como Detectron2 obtiene mejores resultados junto a OsNet, re-identificando correctamente un mayor número de pares. Se puede observar que el número de pares que analiza cada detector es diferente, dando a entender que Detectron2 proporciona un mayor número de predicciones en detección que YOLO, pero, conociendo los resultados del apartado de validación de detección, menos precisos.

De manera general las predicciones de Detectron2 parecen conseguir re-identificaciones ligeramente mejores que YOLO; no obstante, si se observa en proporción, con YOLO se re-identifican correctamente 0.36 de los pares (406/1106) frente al 0.33 de Detectron2 (452/1349)

[1] https://drive.google.com/file/d/1SigwBE6mPdqiJMqhuIY4aqC7–5CsMal/view

Detector	ReID	Máscaras	threshold	pares_ok	pares_totales	personas_mal
YOLOv4	**OsNet**	NO	0.5	343	1106	130
			0.6	387	1106	130
			0.65	406	1106	130
			0.7	376	1106	130
			0.75	232	1106	130
			0.8	106	1106	130
Detectron2	**OsNet**	NO	0.5	351	1349	93
			0.6	409	1349	93
			0.65	452	1349	93
			0.7	425	1349	93
			0.75	263	1349	93
			0.8	123	1349	93
Detectron2	**OsNet**	SI	0.5	98	1349	93
			0.6	103	1349	93
			0.65	97	1349	93
			0.7	96	1349	93
			0.75	93	1349	93
			0.8	59	1349	93

Tabla 4.2: Gráfica de resultados variando el umbral de la similitud del coseno. Las pruebas están realizadas en un conjunto de 40 *frames*.

Se puede observar que el umbral ideal para la similitud del coseno es de 0.65 debido a que es el que maximiza el número de pares re-identificados correctamente en relación al total.

Resultados con máscara de segmentación en detección. Salta a la vista que los resultados empeoran significativamente. Esto se puede deber al hecho de que el modelo de OsNet no está acostumbrado a fondos completamente recortados en los datos de entrada y, por lo tanto, se introduce más ruido del deseado.

4.3.2. Resultados de re-entreno de OsNet

Con los resultados anteriormente mostrados en mente, se decidió buscar mejorar el comportamiento de OsNet. Para ello, se han seleccionado varios modelos de OsNet adicionales al modelo utilizado como base para la validación de re-identificación. En la figura B.1 del anexo B.1 podemos observar que entre estos modelos se tienen:

- **Filas 1 a 3**: Modelo entrenado con varios datasets(*Cross-domain*) y evaluado únicamente en MSMT17.
- **Filas 4 a 5**: Modelo entrenado con solo con MSMT17 y evaluado únicamente en Market1501.
- **Filas 6**: Modelo entrenado y evaluado en el mismo (*Same-domain*) dataset MSMT17.

Para evaluar el funcionamiento de los diferentes entrenamientos, y poder comparar los modelos re-entrenados con los originales, se van a utilizar dos métricas diferentes usadas en los modelos de OsNet proporcionadas en el *Model Zoo*[2]. Estas métricas son estándar en los trabajos relacionados de evaluación de modelos de re-identificación de personas [5]:

- ***Mean Average Precision (mAP)***: precisión media de las predicciones realizadas (los ejemplos seleccionados como más similares) sobre una *query* proporcionada al modelo. En la figura 4.3 el *mAP* sería calculado con los 10 resultados encontrados como más similares a la *query*.
- ***Rank-1***: precisión que indica el porcentaje con el cual la primera etiqueta predicha es la misma que en el *ground truth*. En la figura 4.3 el *Rank-1* sería calculado con solo la primera imagen recuperada.

Es importante destacar que en los modelos originales de OsNet los conjuntos de datos utilizados para entrenamiento no estaban segmentados (contaban con fondo) y en los diferentes re-entrenos realizados se han utilizado los datsets segmentados para tratar de ajustar los modelos a este tipo de datos. También es importante destacar que el trabajo realizado en este experimento ha sido generar un *script* que lance un entrenamiento dados ciertos parámetros: carga de pesos originales, *learning rate*, número de épocas que durará el entrenamiento, cómo se han de muestrear los datos (ej. *random sampling*), etc. Gracias a este *script* se han podido realizar varios experimentos variando ciertos parámetros para tratar de encontrar un modelo de OsNet capaz de ajustarse a imágenes segmentadas.

En la figura B.1 del anexo B.1 se pueden observar los diferentes entrenamientos realizados de *fine-tuning*. Se han realizado todos con la misma configuración y se han extraído *mPA* y *rank-1* de todos ellos para poder compararlos a sus modelos originales.

Como se puede observar, los modelos por lo general empeoran significativamente, aspecto que es común en *fine-tuning* debido a numerosas razones (falta de datos o tiempo). Una de las razones por las cuales pueden verse afectados negativamente estos resultados puede ser que la segmentación de datos (eliminación del fondo) esté generando más ruido del deseado y no sea tan precisa como se espera. Esto se ve mas claramente una vez se imprimen ciertas predicciones realizadas por el sistema para poder ver con más detalle que tipo de imágenes se tienen.

Sin embargo, el segundo modelo re-entrenado mejora significativamente ya que consigue superar los resultados de su versión original con datos de entrada segmentados. No obstante, esto se puede deber a que se ha entrenado con datos de MSMT, los cuales no había visto antes, y una vez aprende de ellos le es más sencillo evaluar en el conjunto de test de este dataset.

En la figura 4.3 y 4.4 podemos observar casos de segmentaciones correctas y de segmentaciones erróneas respectivamente. En el caso de la correcta se aprecia cómo el sistema es capaz de fijarse únicamente en la persona segmentada, ya que el fondo en todas las instancias es negro, facilitando la re-identificación. Por otro lado, en el caso de la incorrecta se puede ver cómo el sistema, que esperaría una persona completa, se fija también en la parte segmentada incorrectamente y

[2] https://kaiyangzhou.github.io/deep-person-reid/MODEL_ZOO

hace que tenga en cuenta las zonas negras para realizar re-identificación, aspecto que se quiere evitar a toda costa.

Para que este paso beneficiara más claramente al sistema de re-identificación seria necesario realizar la segmentación de manera más precisa, para evitar casos en los que se elimine parte de la persona a evaluar.

Figura 4.3: Ejemplo de segmentación de imágenes donde se ha re-identificado correctamente la persona objetivo. A la izquierda se ve la *query* realizada al modelo, y a la derecha los 10 resultados que este ha obtenido para ella (verde si correcto y rojo si incorrecto). Fuente [21].

Figura 4.4: Ejemplo de segmentación de imágenes donde se elimina erróneamente parte de la persona a reconocer.A la izquierda se ve la *query* realizada al modelo, y a la derecha los 10 resultados que este ha obtenido para ella (verde si correcto y rojo si incorrecto). Fuente [21].

Por lo tanto, se ha terminado por utilizar para el sistema base los modelos entrenados originalmente de OsNet, ya que ninguno de los modelos re-entrenados ha podido conseguir resultados que superen al modelo utilizado.

4.4. Evaluación general del sistema

Con el fin de evaluar de manera general el funcionamiento del sistema se decidió grabar un dataset propio en un laboratorio de la EINA. Dicho dataset se creó con la idea de que se asemejara lo máximo posible a un caso normal de uso del sistema, en el que no hubiera grandes multitudes, para que resulte realista el pretender contar el aforo exacto de personas, pero que ocurran numerosas oclusiones, cambios de luz y haya solape entre los campos de visión de las cámaras.

Para observar el comportamiento del sistema, se ha seleccionado un intervalo de 1500 *frames* en los que se conoce en todo momento el número de personas que están ocupando el laboratorio (7 en total) y se compara con las predicciones de aforo que realiza el sistema a lo largo de dicho periodo.

En la figura 4.5 se puede observar un resumen de resultados obtenidos en el intervalo evaluado. Se puede apreciar que el rango de personas predichas como aforo se encuentra entre la franja del 7 al 9. Adicionalmente en la figura 4.6 se puede ver un resumen de el total de *frames* en los que el sistema ha predicho

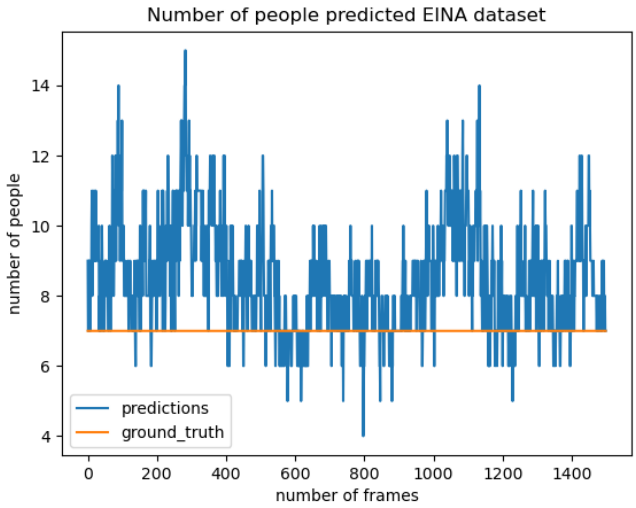


Figura 4.5: Predicciones y *ground truth* sobre el dataset grabado en la EINA.

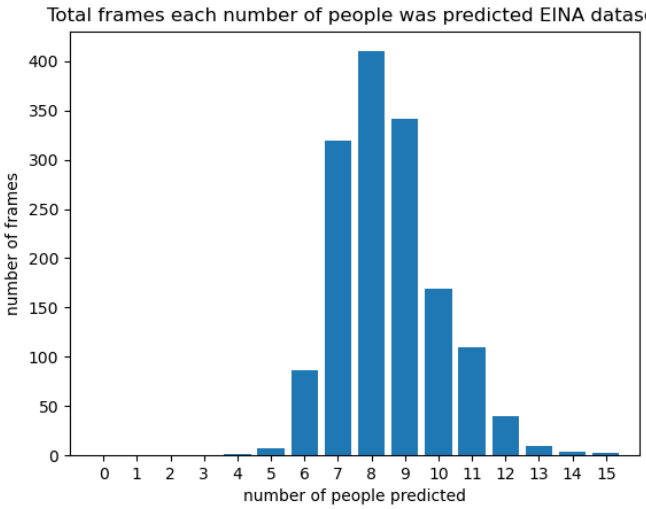


Figura 4.6: Histograma que representa el número de *frames* en los que el sistema ha realizado cada predicción de número de personas.

cada número de personas. Destacando nuevamente el rango común entre el 7 y 9. Esto indica que la precisión del sistema en un entorno donde no haya grandes aglomeraciones y existen numerosos elementos negativos para tareas de re-identificación, es relativamente alta. Ocurren ciertos momentos de alta imprecisión (contado mucho de mas o mucho de menos) pero son momentos muy puntuales y de los que el sistema se recupera rápidamente.

En cuanto a la velocidad del sistema, la cual es dependiente de los tiempos conjuntos de la detección, re-identificación y visualización. Con el uso de YOLO

se alcanza una velocidad de 1 *frame* por segundo (FPS) y con Detectron2 0.25 FPS. Ambos casos son resultados aceptables teniendo en cuenta que en un uso normal no interesa revisar el aforo dentro de un área determinada en todo momento, sino en intervalos de varios segundos.

A continuación se muestran algunos ejemplos extraídos del dataset EINA en los que se puede ver: el correcto funcionamiento con máxima cantidad de gente dentro del laboratorio en la figura 4.8, otro caso de correcto funcionamiento pero con tan solo 4 personas dentro del laboratorio en la figura 4.9 y, por último, un ejemplo de funcionamiento erróneo donde aparecen ciertos problemas de re-identificación en la figura 4.7.

Adicionalmente en el anexo B se pueden encontrar diversas figuras que muestran más ejemplos de otros instantes e intervalos de tiempo de este dataset.

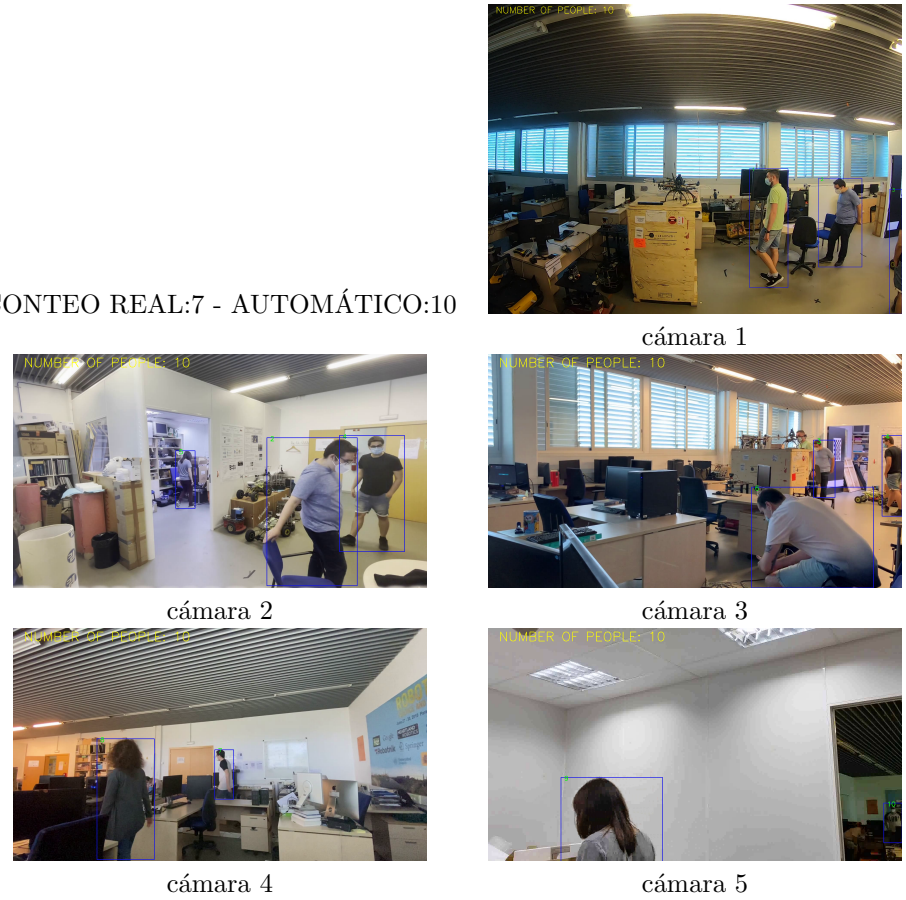


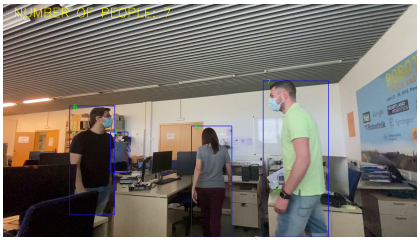
Figura 4.7: Ejemplo del sistema completo funcionando en un instante de tiempo determinado en el EINA dataset. Se pueden apreciar problemas de detección ya que se predicen 10 personas cuando realmente existen 7 dentro del laboratorio. Las cajas azules representan las personas detectadas y en la esquina superior izquierda el número verde fosforito representa el identificador asignado. Un ejemplo de re-identificación errónea ocurre entre la cámara 2 y 5, donde la persona en el interior del cuarto expuesta a dos intensidades de luz, se identifica como distinta entre ambas cámaras. Otro ejemplo idéntico ocurre entre la cámara 4 y 5 con la persona de la camiseta blanca y negra, que aparece mucho más oscura en una de las imágenes, siendo detectada como diferente.

CONTEO REAL:7 - AUTOMÁTICO:7

cámara 1

cámara 2

cámara 3



cámara 4

cámara 5

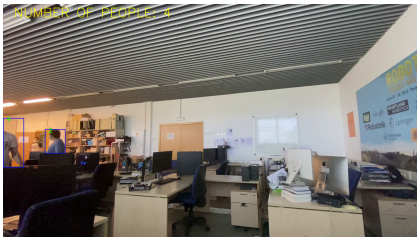
Figura 4.8: Ejemplo del sistema completo funcionando en un instante de tiempo determinado en el EINA dataset. Se puede apreciar, cómo el sistema funciona correctamente pese a encontrar algún problema típico. Las cajas azules representan las personas detectadas y, en la esquina superior izquierda, el número verde fosforito representa el identificador asignado. Un ejemplo interesante es la re-identificación en las cámaras 1, 2, 3 y 5, donde la persona de la camiseta rosada y pantalones cortos es re-identificada correctamente pese a las diferencias de iluminación presentes en las cámaras que la detectan.

CONTEO REAL:4 - AUTOMÁTICO:4

cámara 1

cámara 2

cámara 3



cámara 4

cámara 5

Figura 4.9: Ejemplo del sistema completo funcionando en un instante de tiempo determinado en el EINA dataset antes de que los 7 participantes entraran al laboratorio, concretamente solo con 4 de ellos dentro. En este caso se puede apreciar cómo la persona que entra por la puerta no es detectada en la cámara 4, aspecto que no impide el correcto funcionamiento del sistema, ya que es detectada en la cámara 2.

Capítulo 5

Conclusiones y trabajo futuro

5.1. Conclusiones

Conclusiones técnicas. En este trabajo se han conseguido los objetivos propuestos. Por un lado, se han estudiado diferentes herramientas y algoritmos para la detección y re-identificación de personas. Por otro, se ha diseñado un sistema basado en ellos capaz de controlar el aforo de un área determinada mediante múltiples cámaras. Este sistema cumple todas las funciones previstas: puesta en común de información proporcionada por varias cámaras, detección de personas en dichas imágenes, re-identificación de las personas detectadas, conteo de aforo estimado en áreas de densidad media de manera robusta y, además, sirve como un entorno de aplicación real para evaluar la capacidad de los distintos módulos de cara a funcionar en una aplicación demostrativa. También se han estudiado alternativas con el fin de evaluar mejorías en el rendimiento del sistema y, por último, se ha capturado un dataset original en la EINA para evaluar el sistema.

En cuanto al diseño e implementación del sistema, cabe destacar que se han puesto en marcha e integrado componentes muy diversos, desde modelos de *Deep Learning*, hasta funciones más básicas de tratamiento de imágenes, utilizando bibliotecas muy utilizadas en estos campos como *Pytorch*, *Tensorflow* u *OpenCV*. Como se ha comentado, el sistema es eficiente y, al ser modular, sirve como prototipo para evaluar diferentes bloques para cada una de las tareas principales (detección, re-identificación, evaluación y visualización).

En particular, con el detector y re-id integrado, se han realizado numerosos experimentos que han confirmado que el sistema es capaz de detectar y re-identificar correctamente a las personas encerradas en un área determinada con un nivel de precisión aceptable. El sistema pierde precisión si se somete a datos donde las personas se encuentran en alta densidad o los cambios de iluminación entre cámaras son muy severos, debido a que estos aspectos dificultan las tareas de re-identificación en gran medida. Por otra parte, el sistema es resistente a oclusiones del campo de visión o solapes entre las diversas cámaras, lo cual da flexibilidad a cómo se colocan en el entorno a monitorizar.

Conclusiones personales. De manera más personal, se quiere comentar que este trabajo ha sido una experiencia muy positiva y ha permitido explorar una parte de las herramientas y métodos que existen en el aprendizaje automático. El utilizar dichas herramientas hacia una aplicación realista ha dado más noción a cómo estas pueden usarse no solo para reconocer personas, sino para una gama muy amplia de tareas.

También se ha adquirido experiencia en el uso de modelos de *Deep Learning* que serán importantes para proyectos futuros que se deseen realizar.

Principales problemas encontrados. Se han enfrentado varios problemas de integración, ya que nunca antes se había trabajado con este tipo de modelos de *Deep learning*; no obstante, a medida que se han utilizado más modelos, la experiencia de trabajar con los previos ha facilitado el avance.

Otros problemas encontrados han sido el tiempo a invertir en pre-procesar datasets de gran tamaño (segmentación de máscaras) para obtener las copias de estos, en ocasiones requiriendo de varias horas. Problema idéntico a la hora de entrenar modelos, lo cual ha supuesto un cuello de botella para la realización de experimentos.

5.2. Trabajo futuro

Como trabajo futuro serían interesantes dos aspectos principales.

El primero sería utilizar un método de segmentación más preciso con el que se tuviera seguridad de que las imágenes segmentadas obtenidas nunca eliminarían parte de la persona que se desea reconocer. Con esto se podrían realizar nuevos entrenamientos sobre OsNet con el fin de mejorar la re-identificación con imágenes segmentadas.

El segundo aspecto sería probar el comportamiento del sistema con modelos adicionales, tanto de detección como re-identificación de personas, y poder aumentar el rango de opciones.

Bibliografía

[1] Le Cun Yan, B Yoshua, and H Geoffrey. Deep learning. *nature*, 521(7553):436–444, 2015.

[2] Hong-Yuan Mark Liao Alexey Bochkovskiy, Chien-Yao Wang. Yolov4: Yolov4: Optimal speed and accuracy of object detection. *arXiv*, 2020.

[3] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. `https://github.com/facebookresearch/detectron2`, 2019.

[4] Tatjana Chavdarova, Pierre Baqué, Stéphane Bouquet, Andrii Maksai, Cijo Jose, Timur Bagautdinov, Louis Lettry, Pascal Fua, Luc Van Gool, and François Fleuret. Wildtrack: A multi-camera hd dataset for dense unscripted pedestrian detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5030–5039, 2018.

[5] Kaiyang Zhou, Yongxin Yang, Andrea Cavallaro, and Tao Xiang. Learning generalisable omni-scale representations for person re-identification. *TPAMI*, 2021.

[6] Kaiyang Zhou, Yongxin Yang, Andrea Cavallaro, and Tao Xiang. Omni-scale feature learning for person re-identification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3702–3712, 2019.

[7] Xiaobin Chang, Timothy M Hospedales, and Tao Xiang. Multi-level factorisation net for person re-identification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2109–2118, 2018.

[8] Wei Li, Xiatian Zhu, and Shaogang Gong. Harmonious attention network for person re-identification. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2285–2294, 2018.

[9] Mang Ye, Jianbing Shen, Gaojie Lin, Tao Xiang, Ling Shao, and Steven CH Hoi. Deep learning for person re-identification: A survey and outlook. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[10] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath

Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[11] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[12] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[13] Sumit Saha. A comprehensive guide to convolutional neural networks — the eli5 way.

[14] Prabhu. Understanding of convolutional neural network (cnn) — deep learning.

[15] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[16] Chien-Yao Wang, Hong-Yuan Mark Liao, Yueh-Hua Wu, Ping-Yang Chen, Jun-Wei Hsieh, and I-Hau Yeh. Cspnet: A new backbone that can enhance learning capability of cnn. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 390–391, 2020.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[18] Longhui Wei, Shiliang Zhang, Wen Gao, and Qi Tian. Person transfer gan to bridge domain gap for person re-identification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 79–88, 2018.

[19] Liang Zheng, Liyue Shen, Lu Tian, Shengjin Wang, Jingdong Wang, and Qi Tian. Scalable person re-identification: A benchmark. In *Proceedings of the IEEE international conference on computer vision*, pages 1116–1124, 2015.

[20] Kaiyang Zhou and Tao Xiang. Torchreid: A library for deep learning person re-identification in pytorch. *arXiv preprint arXiv:1910.10093*, 2019.

[21] Longhui Wei, Shiliang Zhang, Wen Gao, and Qi Tian. Person transfer gan to bridge domain gap for person re-identification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.