Departamento de
Informática e Ingeniería
de Sistemas
Universidad Zaragoza
1542

Título: Reidentificación automática de personas en sistemas multi-cámara

Autora: Celia López García

Directora: Ana Cristina Murillo Amal

Codirectora: Sara Casao Martinez

Grado en Ingeniería Electrónica y Automática
Trabajo Fin de Grado

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

Fecha 20/01/2021

Resumen

En los últimos años se está observando un aumento de sistemas reales monitorizados con redes de cámaras. Este tipo de sistemas tienen gran aplicación en el ámbito de la seguridad ya que podemos identificar objetos como por ejemplo personas y conocer su trayectoria y desplazamientos realizados. Otras aplicaciones de re-identificación de personas pueden ser por ejemplo la detección de intrusos o controlar el aforo en ciertos lugares.

Para procesar la gran cantidad de información que nos proporcionan este tipo de sistemas de cámaras, es necesario realizar un procesado automático porque manualmente resulta imposible tratar toda la información. Este trabajo se centra en una tarea muy común en estos sistemas, la re-identificación de personas. En particular se estudian técnicas basadas en aprendizaje profundo (Deep Learning) porque son las que mejor funcionan hoy en día en temas de reconocimiento visual automático.

El objetivo general de este proyecto es el estudio y la evaluación de sistemas existentes de re-identificación de personas y proponer alguna mejora sobre los mismos después de estudiar sus problemáticas principales. En particular este trabajo busca conseguir un sistema que sea más eficiente y generalizable, es decir, obtener que identifique a personas en entornos donde no hayan sido vistos por el sistema con anterioridad.

Después de un estudio de los sistemas de re-idendificación basados en técnicas de aprendizaje profundo (Deep Learning), en particular de las redes neuronales convolucionales (CNN), se ha diseñado un sistema propio mejorado. La mejora implementada consiste en un preprocesado adicional de las imágenes antes de pasar al re-identificador. El sistema utilizado como base para la mejora consta de dos módulos: un módulo de extracción de características de la imagen de la persona a identificar donde se realizará la segmentación de instancias de la imagen, y un módulo de comparación donde comparará esta imagen con las del conjunto de datos seleccionando las más similares e identificando si pertenecen a la misma persona.

Tanto para evaluar los sistemas existentes, como la mejora implementada, se han utilizado métricas estándar y conjuntos de datos públicos. La experimentación se agrupa en tres fases. En el primer experimento evaluaremos los distintos modelos preentrenados de los métodos de re-identificación existentes seleccionados (MobileNet y OsNet), escogiendo las mejores variaciones para los próximos experimentos. En el segundo experimento comprobaremos el funcionamiento de nuestra implementación, combinada con los mejores modelos preentrenados, y comparando los resultados con los del experimento anterior. Para finalizar, en el tercer experimento se entrenan los dos mejores modelos hasta el momento tanto con el conjunto de datos originales y como con la segmentación de instancias de las imágenes del conjunto de datos.

Los resultados obtenidos muestran un estudio exhaustivo de la capacidad (o no) de generalización de los modelos existentes en re-identificación de personas y que la propuesta consigue mejorar estos sistemas, obteniendo un modelo más generalizable como se planteaba al inicio.

Índice general

Capítulo 1

Introducción

1.1. Motivación

Sistemas de monitorización. Cada vez es más frecuente encontrar sistemas reales de monitorización con redes de cámaras. Cuando vamos paseando, observamos como hay cámaras en casi todos los sitios, como en la calle, cajeros, aeropuertos, hospitales, *parkings*... Estos sistema tienen particular relevancia para tareas de seguridad, tanto en el sector público como en el privado.

Estas cámaras generan grandes cantidades de información imposible de procesar de manera manual, por lo cual es necesario conseguir sistemas que analicen automáticamente el contenido de dichas imágenes o vídeos. Una de las tareas que resulta muy relevante en estos sistemas es la re-identificación automática de personas.

Re-identificación de personas. La re-identificación de personas se basa en comparar dos imágenes y decidir si pertenecen o no a la misma persona. Con este proceso lo que se pretende saber es si la persona que aparece en una imagen se corresponde con personas ya vistas anteriormente, almacenadas en el conjunto de datos. En la Figura 1.1 se muestra lo que se busca con la re-identificación de personas.

La re-identificación automática nos permitiría saber si cierta persona ha pasado por un lugar en concreto, si ha realizado cierto recorrido o permitiría realizar control de aforos analizando el número de personas distintas que están pasando por un escenario o están dentro de un establecimiento. En la actualidad, con la problemática del Covid, este tipo de tareas de control de aforo pueden resultar muy útiles.

El proceso de re-identificación en un sistemas multi-cámara plantea numerosos retos, los cuales están relacionados principalmente con la resolución de las siguientes fases o tareas:

- Detección: a través del vídeo de la cámara se analizan todos los marcos de la secuencia de vídeo, detectando la posición en imagen de todas las personas que aparecen en la escena. Por ejemplo, en la Fig.1.2 se ve un ejemplo de detección de personas. Con estas imágenes ya detectadas será con lo que realmente trabajaremos en las siguientes fases.

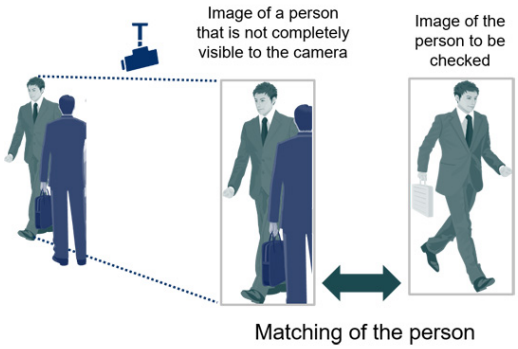


Figura 1.1: Finalidad del proceso de re-identificación de personas. La cámara detecta a una persona parcialmente visible y comprueba si dicha persona ha sido vista con anterioridad, confirmando como efectivamente si que había sido vista (fuente:NEC)

- Descripción: extraer características de la imagen de la persona que nos permitan buscar similares entre el conjunto de datos.
- Re-identificación: predecir si dos imágenes pertenecen a la misma persona.

Este proyecto se centra sobre todo en estudiar las dos últimas fases.

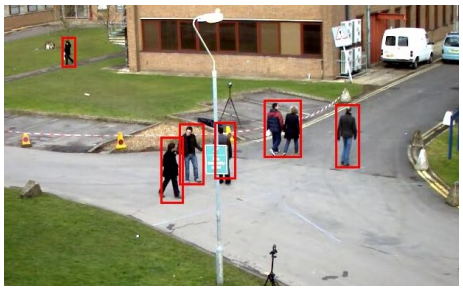


Figura 1.2: Ejemplo de detección de personas en un sistema de monitorización. Las personas detectadas han sido capturadas por un rectángulo rojo. (Fuente: `https://delphimagic.blogspot.com/2013/04/seguimiento-de-personas-animales-y.html`

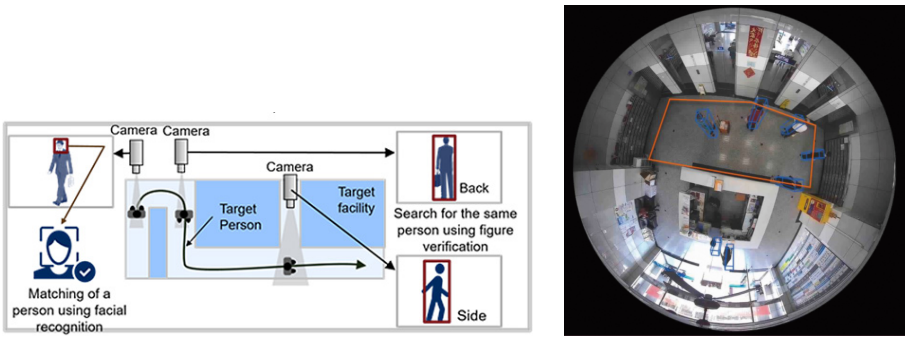
Reconocimiento visual y *deep learning*. Los algoritmos más efectivos desarrollados en el ámbito del reconocimiento visual en los últimos años se basan en técnicas de aprendizaje profundo.

Encontramos ejemplos como [1], donde se discute la aplicación de deep learning a reconocimiento de caras, o en otras aplicaciones como la conducción autónoma [2]. Con la situación que estamos viviendo actualmente, se han utilizado en sistemas de reconocimiento de distanciamiento social, como medida preventiva para covid 19 [3]. Más detalles de estas técnicas mencionadas se explican en el siguiente capítulo.

El proceso de re-identificación no es una excepción. Los trabajos que afrontan esta tarea han demostrado que el uso de técnicas basadas en deep learning son las mas eficaces [4]. Un ejemplo de ello es la aplicación de deep learning para reconocimiento visual de lugares [5]. Por eso, en este proyecto nos vamos a centrar en técnicas de aprendizaje profundo (Deep Learning [6]).

El objetivo principal de este proyecto es el estudio y evaluación de técnicas de deep learning para el análisis y re-identificación automática de personas en sistemas con varias cámaras.

Trabajos relacionados. El problema de la re-identificación automática se ha estudiado mucho, pero sin embargo aun no hay soluciones claramente robustas para funcionar en el mundo real en casos generales. Las grandes compañías van avanzando en este campo, por ejemplo encontramos una solución de NEC donde presentan un sistema que re-identifica personas aunque estén parcialmente ocluidas o giradas [1]. Otro ejemplo similar es el de la compañía VIVOTEC, donde detectan la multitud en tiendas y negocios para identificar cuántas personas hay en un área determinada [2].



(a) Fuente: NEC (b) Fuente: VIVOTEC

Figura 1.3: Ejemplos de aplicaciones comerciales. NEC (a) y VIVOTEC (b).

En el ámbito de la investigación, encontramos numerosas propuestas [7] [8] muchas de ellas, como [4], utilizan una arquitectura del sistema similar al que vamos a implementar en este proyecto: una fase de descripción y una fase de re-identificación.

1.2. Objetivos y Tareas

Reto y Objetivo principal. La problemática actual es que al detectar personas en un entorno determinado, hay que adaptar previamente el método a dicho entorno. Esto crea una pérdida de tiempo y un método poco eficiente, pues sólo funcionará bien en esos entornos donde han sido adaptados. El objetivo que buscamos es conseguir un sistema que intente mejorar dichos aspectos.

[1] https://www.nec.com/en/press/201902/global_20190208_01.html
[2] https://www.vivotek.com/es/learning/feature-article/24/smart-vca

Tareas. Para lograrlo se han definido y realizado las siguientes tareas en este proyecto:

- Estudio de las herramientas básicas para tareas de visión por computador y de las herramientas básicas para tareas de Deep learning.
- Estudio de conceptos básicos de redes neuronales artificiales y Deep Learning
- Estudio de métodos existentes que utilizan técnicas basadas en Deep Learning. Nos centramos en redes neuronales convolucionales.
- Estudio de entornos y datos utilizados para la evaluación de los métodos.
- Selección de 2 algoritmos y puesta en marcha de los mismos.
- Diseño de experimentos para evaluar dichos algoritmos haciendo hincapié en la generalización a distintos entornos. Evaluar y comparar las alternativas seleccionadas.
- Estudio de métodos existentes para la reidentificación de personas poniendo en marcha dos experimentos.
- Implementación de una mejora del sistema.

Para trabajar en el problema de re-identificación de personas se dividió en distintas fases durante el periodo de tiempo de 4 meses. Esto se puede ver en el cronograma de la Figura 1.4.

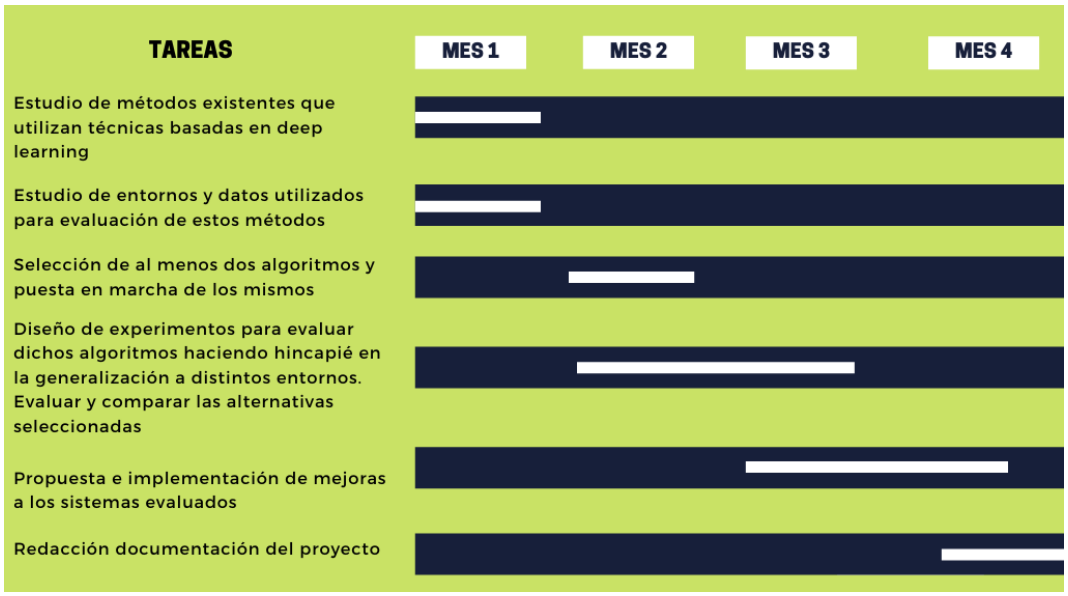


Figura 1.4: División de tareas del proyecto

Metodología y Herramientas. Para llevar a cabo este proyecto, se han estudiado artículos de investigación, código y sistemas de evaluación existentes. Después se ha implementado una variación a dichas propuestas para intentar mejorar los sistemas estudiados. Durante todo este recorrido se ha realizado un seguimiento de reuniones semanales.

Las herramientas utilizadas son bibliotecas de software de computer vision y Deep Learning: OpenCV, Pytorch.

1.3. Contexto

Como paso previo a este proyecto, se han realizado prácticas en el grupo de Robótica, Percepción y Tiempo Real, de la Universidad de Zaragoza en la i3A. En estas prácticas se realizaron tareas previas de estudio de ideas y técnicas más básicas y generales de aprendizaje automático y Deep Learning (estudio de herramientas, los conceptos básicos de redes neuronales convolucionales y la implementación de algoritmos de reconocimiento automático existentes en el grupo de investigación para aprender y evaluar su funcionamiento).

1.4. Resumen de contenido de la memoria

El proyecto está dividido de la siguiente manera:

- En el Capítulo 1, se presenta la motivación y objetivos que se pretenden cumplir.
- En el Capítulo 2 se explica el trabajo relacionado en detalle.
- En el Capítulo 3 se describe el sistema propuesto para obtener mejoras en los sistemas de re-identificación estudiados.
- En el Capítulo 4 se describen los experimentos realizados para validar el funcionamiento de los métodos existentes en re-identificación y del sistema propuesto de mejora.
- En el Capítulo 5 se describen las conclusiones obtenidas de la realización del proyecto junto con algunos trabajos futuros.

Capítulo 2

Re-identificación de personas con métodos de deep learning

2.1. Definición del problema de re-identificación.

El problema de re-identificación consiste en lo siguiente: dada una imagen capturada por el sistema de monitorización, donde se detecta a la persona que aparece en ella, averiguar si esa persona pertenece a alguna de las personas vistas anteriormente por el sistema.

Este problema no es fácil de resolver, ya que influyen muchos aspectos como por ejemplo la iluminación del ambiente donde se realiza la foto, o la postura de la persona.

En la actualidad, la mayoría de los métodos usados para la re-identificación son generalmente adaptados con anterioridad al entorno para que funcionen de manera más robusta. Uno de los objetivos que buscamos con este proyecto es conseguir un sistema que generalice lo mejor posible a datos que no haya visto antes, es decir, un algoritmo con el que no sea necesario la adaptación al entorno para conseguir buenos resultados.

Los métodos existentes que mejor funcionan hoy en día para la re-identificación de personas están basados en técnicas de Deep Learning [6], conocidas en castellano como aprendizaje profundo. Como se ha mencionado, este proyecto centra el estudio en este tipo de técnicas. Una característica clave de estas técnicas es que no tienen que estar provistos de características 'hechas a mano', sino que aprenden las características más adecuadas para el problema a partir de los datos de la imagen.

2.2. Conceptos básicos de *Deep Learning*

El Deep Learning es un subcampo específico del aprendizaje automático (*Machine Learning* [9]) que trabaja con redes neuronales artificiales profundas para obtener una predicción dado un conjunto de entradas. Este tipo de redes neuronales implican capas sucesivas de representaciones, que se aprenden au-

tomáticamente a partir de la exposición de los datos de entrenamiento. Cada una de estas capas están formadas por un conjunto de unidades, llamadas neuronas, conectadas entre sí. En la figura 2.1 podemos ver un ejemplo de red neuronal donde cada nodo circular representa una neurona y cada flecha representa la conexión de entrada y salida entre ellas. La arquitectura de las redes neuronales está formada por tres bloques de capas:

- Capa de entrada: donde se le pasan los datos de entrada, en nuestro caso imágenes.
- Capas ocultas: realizan los cálculos matemáticos. Suelen estar formadas por múltiples capas.
- Capa de salida: es el último eslabón, devuelve la predicción realizada.

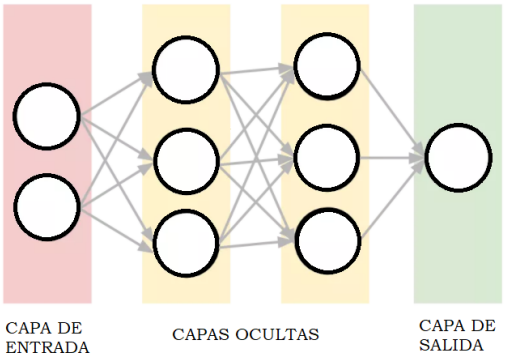


Figura 2.1: Ejemplo de esquema de red neuronal. Cada nodo circular representa una neurona y cada flecha representa la conexión de entrada y salida entre ellas.

Vamos a entrar más en contexto en cómo funcionan las capas de las redes. La cantidad total de capas en la red es lo que le da la profundidad al modelo. Cada capa tiene asociado un peso (conjunto de números). Estos pesos se actualizan en el entrenamiento a medida que se va recorriendo la red, pero una vez terminado el entrenamiento los valores quedan fijos, no cambian. Cuando introducimos datos de entrada a la red (en nuestro caso imágenes), a medida que va recorriendo las distintas capas que forman la red, estos valores se van transformando debido a los pesos de dicha capa. Si esta capa recorrida es la última capa, con estos nuevos valores se procederá a la clasificación, devolviendo la predicción, pero en caso contrario estos valores serán los de entrada de la siguiente capa, realizándose el mismo proceso que antes pero con los pesos de la capa que recorre.

En la figura 2.2 podemos ver un ejemplo de modelo neuronal con n entradas y sus pesos. La operación que se realiza en una neurona es la siguiente:

$$y = f(\mathbf{WX}), \tag{2.1}$$

donde W son los pesos de la capa, X son los datos (valores) de entrada a la capa

Para conseguir que una red aprenda a reconocer el contenido de las imágenes, necesitamos entrenarla. Entrenar consiste en ajustar cada uno de los pesos de las entradas de todas las capas de la red neuronal. Es decir, ajustar las representaciones de las clases de la red para que a la hora de detectar objetos

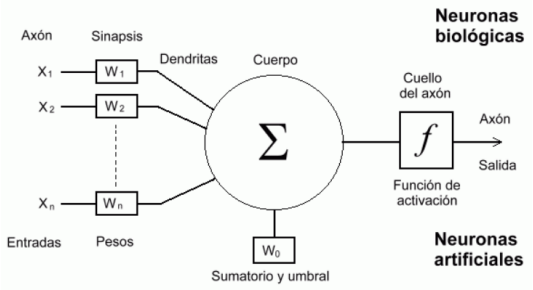


Figura 2.2: Ejemplo de modelo de una neurona, con entrada de dimensión n. Se compara la terminología de redes neuronales artificiales con la de las biológicas. (Fuente: `http://www.cs.us.es/~fsancho/?e=72`)

en la imagen que le pasamos pueda comparar las representaciones e identificar los objetos correctamente. Este entrenamiento se debe hacer con una gran cantidad de imágenes de contenido conocido para que la red pueda captar sus características únicas.

Hay muchos tipos de redes neuronales artificiales (FNN, RNN, PNN, etc.), pero este trabajo se centra en las Convolutional Neural Network (CNN), conocidas como Redes Neuronales Convolucionales, que son las más habituales en clasificación de imágenes. Las CNN son un tipo de redes neuronales artificiales formadas por múltiples capas cuya operación clave es la convolución.

Estructura de una CNN: Para entender mejor la arquitectura de la CNN y tener una representación visual, en la Figura 2.3 se ve un pequeño esquema con las principales capas que constituyen una red neuronal convolucional y que explicaremos mas adelante.

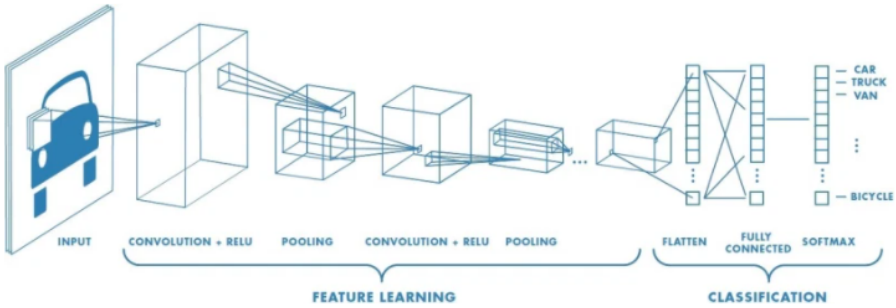


Figura 2.3: Ejemplo de una arquitectura de red CNN. (fuente: `https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53`)

Las redes neuronales convolucionales consisten en múltiples capas de filtros convolucionales de una o más dimensiones. En la entrada de la red (capa de entrada) introducimos una imagen, que en realidad es una matriz de píxeles. Las imágenes que manejamos son a color, donde el valor de cada píxel se obtiene

como la combinación de tres valores que corresponden a tres canales (RGB). Estos valores de píxel tienen un rango de 0 a 255.

Al trabajar con imágenes a color, antes de alimentar la red, hay que normalizar los valores para que nos queden entre 0 y 1. Esto se hará con una simple transformación "valor/255". La red toma los píxeles de la imagen como entrada. Una vez tenemos trasformada la imagen, la red puede empezar a trabajar.

Las principales capas que forman una red neuronal convolucional son las siguientes:

- **Capa convolucional**: este tipo de capa es distintivo respecto a otros tipos de redes. Las convoluciones consisten en tomar grupos de píxeles cercanos de la imagen de entrada e ir operando con ellos (mediante producto escalar) contra una pequeña matriz llamada *kernel*. Esta matriz recorre todas las imágenes de entrada (de izquierda a derecha, de arriba a abajo) y genera una nueva matriz de salida, que será la entrada a la siguiente capa. En la figura 2.4 podemos ver una representación visual de como realiza el proceso de convolución con matriz kernel.

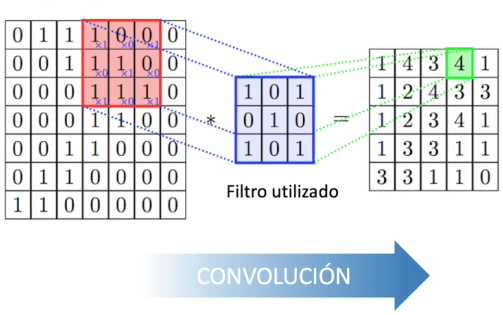


Figura 2.4: Representación visual del proceso de convolución en red CNN. Vemos como se aplica un kernel o filtro de $3x3$ a una imagen (matriz de pixeles) con un *step* de 1 (kernel se mueve de uno en uno) y *stride* 1 (se deja un margen de 1). (Fuente: `https://www.diegocalvo.es/red-neuronal-convolucional/`).

- **Capa de activación**: esta capa suele ir después de una capa de convolución. Aplica la función de activación generando límites de decisión no lineales.
- **Capa Pooling o de agrupación**: es habitual añadirla entre cada capa de convolución. Se encarga de reducir el tamaño espacial de la representación para reducir la cantidad de parámetros y cálculos en la red prevaleciendo las características más importantes detectadas por el filtro. En la Figura 2.5 vemos una representación de este proceso, en particular, un ejemplo de *max-pooling*.
- **Fully Connected**: este tipo de capa normalmente se encarga de la clasificación. Puede haber una o varias y consolidan en un vector la salida de la última capa convolucional (también conocido como el último mapa de características extraídas por la red). En la última capa se realiza la clasificación final, y tendrá tantas neuronas como número de clases a predecir.

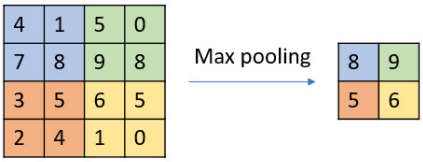


Figura 2.5: Reducción de tamaño mediante capa *max-pooling*. El pooling se aplica de regiones de $2x2$ y consiste en conservar solamente el valor máximo de cada una de esas regiones, por lo tanto, como se puede ver, pasamos de una imagen $4x4$ a una imagen $2x2$. (Fuente: `https://www.ellaberintodefalken.com/2019/10/vision-artificial-redes-convolucionales-CNN.html`)

Las salidas de las diferentes capas intermedias de las CNN se pueden interpretar como descriptores de la imagen que se le pasa a la red. En nuestro caso, es el punto clave de la CNN que se va a utilizar para la re-identificación.

2.3. CNN aplicadas a re-identificación

Como se ha comentado, la re-identificación de personas no es una misión fácil de conseguir. Influyen aspectos como por ejemplo la iluminación, perspectiva, escala o resolución entre las mismas. Tenemos que tener en cuenta que se producen grandes variaciones intra-clases (instancia/identidad) debido a los cambios en las condiciones de visualización de la cámara. No siempre vamos a encontrarnos con una imagen dónde se vea la persona completamente, con toda su vestimenta o complementos, hay veces que solo se ve una parte de la persona porque la tapa un coche o un semáforo por ejemplo. Esto se produce por el cambio de vista a través de cámaras donde produce grandes cambios de apariencia en el área de ciertas partes de la persona, lo que dificulta la identidad de la misma persona. Otro aspecto a tener en cuenta son las pequeñas variaciones entre clases porque la gente suele llevar ropa similar. Desde las cámaras, con distancia, se pueden ver casi iguales. En la Figura 2.6 podemos ver cómo influyen los inconvenientes mencionados anteriormente.

Figura 2.6: Ejemplos de imágenes difíciles de re-identificar. Los tres chicos de la derecha tienen la vestimenta junto a la mochila prácticamente igual, lo único que varía es el dibujo de la camiseta, estos pequeños detalles son los que a la red le es difícil diferenciar. Con las chicas pasa lo mismo, le resulta costoso diferenciarlas debido a su vestimenta y corte de pelo. (Fuente: [4])

En los sistemas de re-ID el objetivo es aprender características discriminatorias pero a su vez invariantes a este tipo de variaciones intra-clase. Para identificar a la persona correctamente, sin confundirla con impostores nos fijamos en todas las características tanto regiones locales pequeñas (tipo de zapatillas, tamaño de mochila...) como regiones globales (sexo de la persona, colores de la ropa...). Las redes convolucionales profundas (CNN) han sido diseñadas para tareas de reconocimiento a nivel de categoría de objeto que son fundamentalmente diferentes de la tarea de reconocimiento a nivel de instancia en ReID.

2.3.1. Métodos para Re-identificación de personas basados en CNNs.

Los sistemas de re-identificación mas efectivos hoy en día son los basados en Deep Learning y están formados por redes neuronales convolucionales (CNN).

En particular, en este trabajo nos centramos en el entorno para re-identificación *Torchreid*, desarrollado en [10, 4, 11][1], donde se estudian y comparan numerosas alternativas para re-identificación.

El sistema de base de dicho entorno que se estudia en este trabajo consiste en dos fases: 1) Fase de descripción y 2) Fase comparación de las personas detectadas, este esquema es el que se sigue también en el sistema propuesto en este trabajo, detallado en el capitulo siguiente.

2.3.1.1. Fase de descripción

Dentro de *Torchreid* se trabaja con diferentes modelos para describir una imagen de una persona. En particular, los mas relevantes seleccionados y utilizados en este trabajo son los siguientes:

- MobileNetV2 [12]: Esta arquitectura se ha elegido porque es conocida por su eficiencia y rapidez en tareas de clasificación de imagen genéricas. MobileNet se basa en una estructura residual invertida donde la entrada y la salida del bloque residual son capas delgadas de cuello de botella opuestas a los modelos residuales tradicionales que utilizan representaciones expandidas en la entrada. MobileNetV2 utiliza convoluciones en profundidad ligeras para filtrar características en la capa de expansión intermedia. Además, se eliminaron las no linealidades en las capas estrechas para mantener el poder de representación. La arquitectura de este modelo se puede ver en la Tabla 2.1.

- OSNet [4]: Esta arquitectura se ha elegido porque es una nueva CNN propuesta específicamente para el problema de re-identificación. Logra el aprendizaje de funciones *omnidireccionales* diseñando un bloque residual compuesto por múltiples flujos convolucionales, cada uno de los cuales detecta características a una determinada escala. Este método introduce una nueva puerta de agregación unificada para fusionar dinámicamente características de múltiples escalas con pesos por canal dependientes de la entrada. Para aprender de manera eficiente las correlaciones de canales espaciales y evitar el sobreajuste, el bloque de construcción utiliza convoluciones puntuales y en profundidad. Al apilar dichos bloques capa por

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

Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	

Tabla 2.1: Arquitectura general de MobileNetV2

capa, el método OSNet es ligero y se puede entrenar desde cero en los puntos de referencia de ReID existentes. La arquitectura de este modelo se puede ver en la Figura 2.7. OSNet-AIN[11] es una variación de OSNet que también se considera, cuya principal modificación es la estructura de bloques, ya que utiliza bloques de aprendizaje de funciones a escala omnidireccional con normalización de instancias.

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 2.7: Arquitectura del método OSNet (fuente: [4])

El entorno *Torchreid* facilita diferentes variaciones de estos modelos ya entrenadas con las siguientes configuraciones.

Loss function. Han sido entrenados utilizando la función **Softmax**. Esta función se encarga de controlar la salida de la red neuronal. Para entrenar, hay que calcular una función de perdida en la salida, que nos indica qué tan lejos está la salida predicha de lo que se esperaba. La *SoftMax loss* esta basada en el cálculo de la cross-entropy loss, y se calcula como define la Ecuación 2.2:

$$\sigma : \mathbb{R}^K \to [0,1]^K$$
$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}, \qquad j = 1, \ldots, K, \tag{2.2}$$

donde z es el vector de dimensiones K.

Optimizador. También han usado en el entrenamiento el optimizador **Adam** (Adaptive moment estimation) [13]. Este algoritmo es de los más recientes y está construido en base a dos optimizadores (AdaGrad y RMSProp), combinando sus mejores características. Se mantiene un factor de entrenamiento por parámetro y además de calcular RMSProp, cada factor de entrenamiento también se ve afectado por la media del momentum del gradiente. Su función es ajustar un poco el valor de las ponderaciones, en una dirección que reducirá la puntuación de pérdida utilizando la puntuación de distancia como señal de retroalimentación.

Por último, la re-identificación de la persona se realiza según las distancias calculadas entre los descriptores de la imagen que estamos intentando identificar y el conjunto de datos de referencia. Este cálculo lo vamos a hacer mediante la distancia métrica euclídea. Con las distancias ya calculadas, la persona más parecida a la que estamos identificando será la que menor distancia tenga.

Una vez entrenados los modelos, realmente la descripción de una imagen se consigue extrayendo las características de la capa *fully conected* que se encuentran antes de la última capa de salida.

2.3.1.2. Fase de comparación

Una vez que están calculados los descriptores para representar las imágenes de personas, el siguiente paso para conseguir la re-identificación es encontrar si la persona detectada se encuentra en la base datos.

Para ello, se utiliza el algoritmo de *nearest neighbour* para encontrar las imágenes de la base de datos más similares respecto a la que queremos identificar. Para conseguirlo se realiza haciendo el cálculo de distancias de los descriptores de las imágenes. Con las distancias ya calculadas, para identificar a la persona nos fijamos en la distancia más cercana siendo esa la persona más parecida o similar a la que estamos identificando.

Capítulo 3

Sistema de re-identificación implementado

Descripción del sistema: En este capítulo se va a describir el sistema propuesto para mejorar los sistemas de re-identificación estudiados. La idea global de la propuesta se muestra en la Fig. 3.1. En primer lugar, la entrada al sistema es una imagen recortada de la persona a re-identificar (Img). A continuación, se obtiene un descriptor de dicha imagen mediante el bloque Módulo de descripción de imagen (p_i). Finalmente, con el Módulo de comparación obtenemos la persona de la base de datos (BD) con mayor similitud a Img.

El objetivo es encontrar a la persona de la imagen (Img) en las imágenes almacenadas de la base de datos (BD) pertenecientes a la misma persona.

En nuestro caso, asumimos que en la base de datos se encuentran todos los ejemplos de personas que vamos a querer re-identificar, es decir, la persona a re-identificar ya ha pasado anteriormente por el sistema.

A continuación se explica en que consiste cada módulo.

3.1. Módulo de Descripción

Este módulo recibe una imagen recortada de la persona a re-identificar Img y devuelve un descriptor de dicha persona, p_i. Este descriptor es un vector de características que se obtiene mediante una red neuronal convolucional (CNN).

La mejora que se propone evaluar en este trabajo es un paso de pre-procesado que consiste en segmentar la imagen de entrada, desechando así, la información que no es relevante para nuestro proceso. Con esto buscamos facilitar el trabajo a la red, dejando lo esencial.

La segmentación es un proceso de clasificación por píxel que asigna una categoría a cada píxel de la imagen analizada. Es una técnica similar a lo que los humanos solemos hacer cuando miramos algo, identificar que partes de la imagen pertenecen a una etiqueta o categoría. En la actualidad, la segmentación tiene muchas aplicaciones como la conducción autónoma, la segmentación facial, la segmentación de objetos en interiores, Geo Land Sensing, etc. Este proceso es todo un desafío porque además de la detección correcta de todos los objetos en una imagen también requiere la segmentación precisa de cada instancia.

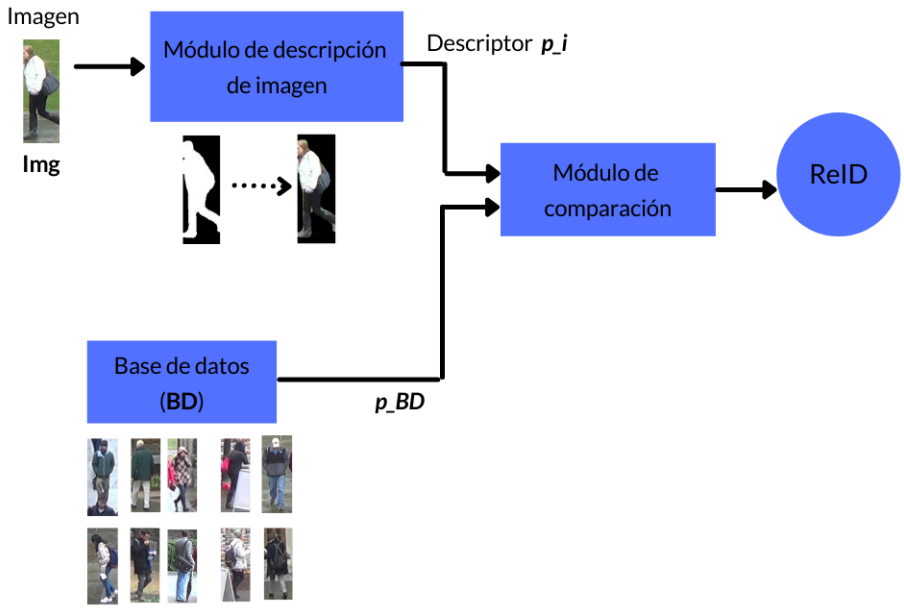


Figura 3.1: Esquema del modelo de Re-identificación implementado formado por dos módulos principales (Módulo de descripción de imagen y Módulo de comparación). Como entrada al sistema se mete una imagen recortada de una única persona. La salida es la imagen más similar a la de entrada ubicada en la base de datos.

Existen distintos tipos de segmentación, pero en este proyecto se va a usar la segmentación de instancias. Mientras que la detección de objetos estándar se basa en definir el área donde se encuentran los objetos con "*bounding boxes*", la segmentación de objetos consiste en definir el contorno de un objeto de la manera más precisa posible. En la Figura 3.2 se puede ver un ejemplo de la diferencia entre ambos. La segmentación de instancias es el proceso mediante el cual se detecta cada objeto en la escena y se genera una máscara individual para cada detección, permitiendo así, extraer el objeto detectado con mayor precisión. Este método, puede entenderse como la combinación de dos procesos. En primer lugar, la detección del área rectangular que contiene el objeto y luego, la obtención de la máscara que segmenta dicho objeto.

Para hacer la segmentación de instancias usamos el método Mask R-CNN [14]. En la Figura 3.3 se puede observar su estructura. Este método es una extensión de la red de detección Faster R-CNN[15] a la que se le añade una rama en paralelo a la ya existente de predicción de "*bounding boxes*". Este complemento realiza la predicción de máscaras para cada uno de los objetos que aparecen en la escena. Una de las ventajas de este método es la mejora en la precisión de la máscara conseguida gracias a la capa simple que se añade llamada RoIAling. Esta capa sirve para corregir la desalineación mejorando la

(a) (b)

Figura 3.2: Ejemplo de detección de personas de dos maneras: (a) mediante detección de objetos estándar (indicando solamente la ventana de la imagen, en verde, donde se encuentra la persona) y (b) mediante segmentación de instancias (indicando los píxeles exactos de la imagen que pertenecen a cada persona, marcados en rosa).

precisión de la máscara en un relativo 10 % a 50 %, mostrando mayores ganancias bajo métricas de localización más estrictas.

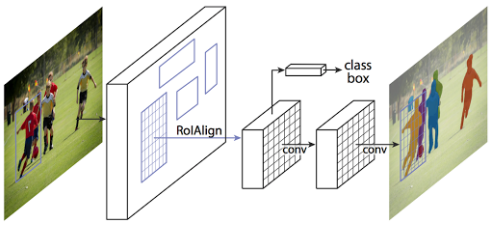


Figura 3.3: Estructura del proceso de segmentación de Mask R-CNN (fuente: [14]

El modelo utilizado para hacer la segmentación es el FCN-ResNet101 preentrenado. Está formado por un modelo de red neuronal totalmente convolucional(CNN) denominado ResNet-101. Los modelos previamente entrenados han sido entrenados en un subconjunto de COCO train2017, en las 20 categorías presentes en el conjunto de datos de Pascal VOC. En nuestro caso, nos centraremos en las máscaras etiquetadas como persona.

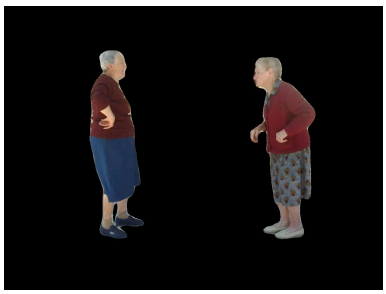
Una vez que hemos obtenido las máscaras correspondientes a la etiqueta deseada, realizamos una operación binaria mediante el método de Opencv *cv*2.*bitwise_and*() para conseguir la segmentación de la imagen original. En la Fig.3.4 podemos ver el resultado final de la segmentación de instancias.

Por lo tanto, este módulo cuenta con dos versiones. La primera, en la que las imágenes de entrada serán las originales, y una segunda en la que dichas imágenes habrán tenido un pre-procesado de segmentación de instancias.

En cuanto a la extracción de características se plantean dos opciones:

- Utilizar modelos preentrenados explicados previamente en la Sección 2.3.1.1.
- Utilizar modelos entrenados por nosotros. La principal modificación se centra en la base de datos de entrenamiento, la cual se ha segmentado

(a) Imagen real de entrada



(b) Imagen final segmentada

Figura 3.4: Ejemplo final del resultado de hacer la segmentación de instancias de la imagen junto a la operación binaria. La imagen (a) es la de entrada y la imagen (b) es la que nos devuelve al hacer la segmentación de instancias y la operación binaria.

para intentar obtener una mejora en la generalización en los modelos.

Todos los modelos se han obtenido de la biblioteca *torchreid*: `https://kaiyangzhou.github.io/deep-person-reid/MODEL_ZOO.html`

3.2. Módulo de comparación

En este módulo la entrada es el descriptor (o vector de características) de la imagen a identificar, p_i, y los descriptores de las imágenes de la base de datos, p_BD. La función de este módulo es encontrar una persona de la base de datos con características las más similares a la que queremos identificar.

Para ello, como se ha descrito en la sección 2.3.1.2, se utiliza el algoritmo de *Nearest Neighbour*. Este algoritmo se basa principalmente en el cálculo de las distancias entre los descriptores, y la decisión se toma de acuerdo a la distancia mínima. Es decir, para seleccionar a la persona con mayor similitud de la base de datos, miramos las distancias de cada imagen, y la que menor distancia tenga con respecto a la imagen a identificar, será la escogida.

En particular, en el sistema implementado se usa la distancia Euclídea (ecuación 3.1), donde P y Q son puntos del espacio euclídeo n-dimensional.

$$P = (p_1, p_2, \ldots, p_n) \qquad Q = (q_1, q_2, \ldots, q_n)\,,$$
$$d_E(P,Q) = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2}. \tag{3.1}$$

Capítulo 4

Evaluación de los métodos de re-identificación

Este capítulo muestra los experimentos realizados para analizar los diferentes modelos para re-identificación estudiados y evaluar la modificación propuesta.

4.1. Configuración de Experimentos

4.1.1. Datos utilizados

Todos los conjuntos de datos públicos utilizados en estos experimentos consisten en un conjunto de imágenes capturadas en un entorno en concreto. Las imágenes consisten en imágenes recortadas donde cada imagen contiene solo una persona. Todos están divididas en 3 sub-conjuntos: entrenamiento, galería y consulta.

- Imágenes de entrenamiento: son las que se utilizan para entrenar la red y conseguir así características lo suficientemente descriptivas como para poder realizar correctamente la identificación de las personas.
- Imágenes de galería (BD): conjunto de imágenes etiquetadas de las personas conocidas por el sistema.
- Imágenes de consulta (Img): imagen de la persona a re-identificar, es decir, imagen a comparar con las que se encuentran en la de galería.

Las bases de datos que vamos a utilizar en nuestros experimentos son:

- **DukeMTMC-Re-ID [16]** (**Duke**): formado por imágenes de vigilancia realizadas en el campus de la Universidad de Duke en 2014 capturadas mediante 8 cámaras. Contiene un total de 36411 imágenes de 1404 personas. Estas imágenes están divididas en: 16522 de entrenamiento, 2228 de consulta y 17661 de galería. En la Figura 4.1 podemos ver algunos ejemplos de las imágenes de este conjunto de datos, en concreto estas imágenes pertenecen a las imágenes de consulta.
- **Market1501 [17]** (**Market**): este conjunto de datos han sido recopilados frente a un supermercado en la Universidad de Tsinghua mediante

Figura 4.1: Ejemplos de algunas imágenes pertenecientes al conjunto de datos de Duke, en concreto estas imágenes son ejemplos de imágenes de consulta.

6 cámaras. Lo forma un total de 32217 imágenes de 1501 personas. Las imágenes están divididas en: 12936 de entrenamiento, 3368 de consulta y 15913 de galería. En la Figura 4.2 podemos ver algunos ejemplos de las imágenes de este conjunto de datos, en concreto estas imágenes pertenecen a las imágenes de consulta.

Figura 4.2: Ejemplos de algunas imágenes pertenecientes al conjunto de datos de Market, en concreto estas imágenes son ejemplos de imágenes de consulta.

4.1.2. Métricas utilizadas

Las métricas que utilizaremos para la evaluación del modelo son las siguientes:

- *Rank-N*: representa el porcentaje de aciertos que hay en las N imágenes de la galería con menor distancia a la imagen de consulta. Como ya comentamos en la sección 3.2, el cálculo de esta distancia se realiza mediante la distancia euclídea. Para los resultados de las evaluaciones en los modelos preentrenados se ha seleccionado el valor de Rank1 ($N = 1$), y para los modelos que entrenamos desde cero se han obtenido los valores de Rank1, Rank5 y Rank10, con el objetivo de tener un análisis más completo del sistema.
- Mean Average Precision (mAP): conocida como la media de la precisión media. Una vez realizado el ranking de imágenes de la galería (BD), esta métrica informa sobre lo cercanos que se encuentran los ejemplos de BD correspondientes con la identidad de la imagen de consulta. Se obtiene mediante la Ecuación 4.1, donde Q es el número de consultas en el conjunto y $AveP(q)$ es la precisión media para una consulta determinada, q.

$$mAP = \frac{\sum_{q=1}^{Q} AveP(q)}{Q} \tag{4.1}$$

4.1.3. Métodos comparados

Los modelos que vamos a comparar pertenecen a los métodos estudiados y explicados en el Capítulo 2, y son variaciones de los modelos MobileNet [12] y OsNet [4].

Provenientes del método MobileNet consideramos las siguientes variaciones:

- *mobilenetv2_x1_0*: este modelo es el MobileNet original. No tiene ninguna variación ya que el factor de multiplicación es de 1, las dimensiones de los canales, características y capas quedan igual.
- *mobilenetv2_x1_4*: en esta variación la estructura del modelo se modifica por un factor de 1.4. Esta modificación produce el aumento de dimensión de canales (32*1.4) y características (1280*1.4), incluyendo también la modificación de las distintas capas.

Del método OSNet se consideran las siguientes variaciones del modelo:

- *osnet_x1_0*: está compuesta por 1000 clases, tres bloques de aprendizaje de funciones omni-escala y dimensiones de canales de filtro [64, 256, 384, 512].
- *osnet_x0_75*: tiene la misma estructura que *osnet_x1_0* exceptuando los canales que están reducidos un 0.75, quedándose en [48, 192, 288, 384].
- *osnet_x0_5*: tiene la misma estructura que *osnet_x1_0* exceptuando los canales que están reducidos un 0.5, quedándose en [32, 128, 192, 256].
- *osnet_x0_25*: tiene la misma estructura que *osnet_x1_0* exceptuando los canales que están reducidos un 0.25, quedándose en [16, 64, 96, 128].
- *osnet_ibn_x1_0*: tiene la misma estructura que *osnet_x1_0* pero con la ampliación de una capa IBN.
- *osnet_ain_x1_0*: esta variación es igual a *osnet_x1_0* con la diferencia de que en la estructura de bloques también tiene bloques de aprendizaje de funciones a escala omnidireccional con normalización de instancias.

Utilizaremos la versión pre-entrenada de todos estos modelos disponible en el entorno *torchreid* [1]. Las versiones más prometedoras también las utilizamos para realizar entrenamientos desde cero como se describe en el Experimento 3.

En la Tabla 4.1 se ven las características de memoria y velocidad de todas estas variaciones.

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

Modelo	PARÁMETROS (10^6)	GFLOPs
mobilenetv2_x1_0	2.2	0.2
mobilenetv2_x1_4	4.3	0.4
osnet_x1_0	2.2	0.98
osnet_x0_75	1.3	0.57
osnet_x0_5	0.6	0.27
osnet_x0_25	0.2	0.08
osnet_ibn_x1_0	2.2	0.98
osnet_ain_x1_0	2.2	0.98

Tabla 4.1: Características de los distintos modelos de descripción de imagen estudiados.

4.2. Experimento 1: Evaluación de todos los modelos

Objetivo: En esta parte de la experimentación se quiere verificar el funcionamiento de los métodos existentes de re-identificación cuando se entrena y testea en conjuntos de datos distintos. Con esto buscamos la selección de los modelos que mejor generalizan para posteriores pruebas.

Descripción experimento: Los modelos con los que trabajamos en este experimento ya han sido previamente entrenados. Lo primero que hacemos es testear con el mismo conjunto de datos con el que ha sido preentrenado. De esta forma, obtenemos la referencia del valor máximo que se puede conseguir con ese modelo. Después realizamos una evaluación cruzada testeando el mismo modelo en otro conjunto de datos.

Resultados: En la Figura 4.3 podemos ver ejemplos obtenidos de la evaluación de los mejores modelos seleccionados pre-entrenados en *Duke* y testeados en *Market*.

En el apéndice A hay más ejemplos y detalle de todos los resultados englobados en este experimento 1. A continuación se muestran y discuten los resultados más relevantes.

En la tabla 4.2 podemos ver los resultados de entrenar los modelos en el conjunto de datos de *Market*, y en la tabla 4.3 los resultados de entrenar en *Duke*.

A la hora de elegir el mejor modelo es necesario tener en cuenta no sólo los resultados de Rank1 y mAP, sino también el número de parámetros y GFLOPs de cada uno. Estas características se pueden ver en la tabla 4.1.

Con esto presente podemos empezar a comparar resultados. Echando un primer vistazo, se ve claramente como los modelos de *OSNet* dan mejores resultados que los de *MobileNet*. El modelo con mejor resultado es *osnet_ain_x*1_0, por lo que será el primero que escogeremos. También seleccionaremos *osnet_ibn_x*1_0 al ser el segundo mejor y tener las mismas características.

Como hemos comentado, influyen las características del modelo, por eso el siguiente que escogeremos será el que tenga menor tamaño y sea más rápido. Como se observa en la tabla 4.1, dicho modelo equivale a ser *osnet_x*0_25.

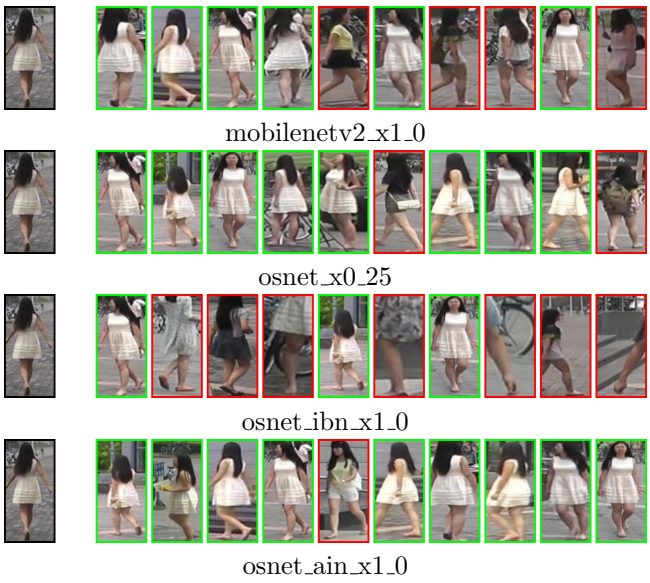


Figura 4.3: Experimento 1: Ejemplos de la evaluación en los mejores modelos seleccionados. En cada fila, la imagen de la izquierda es la imagen de *consulta*, y el resto son las imágenes más similares encontradas (de izquierda a derecha) por el sistema. El cuadro verde indica acierto mientras que rojo indica error. Estos modelos están preentrenados en *Duke* y testeados en *Market*.

Modelo	Train	Test	Rank1 (mAP)
mobilenetv2_x1_0	**Market**	**Duke**	**25.1 (13.3)**
mobilenetv2_x1_0	**Market**	**Market**	**86.2 (68.6)**
mobilenetv2_x1_4	Market	Duke	23.1 (12.7)
mobilenetv2_x1_4	Market	Market	86.8 (69.9)
osnet_x1_0	Market	Duke	34.6 (19.7)
osnet_x1_0	Market	Market	94.2 (83.6)
osnet_x0_75	Market	Duke	33.5 (18.8)
osnet_x0_75	Market	Market	93.9 (82.4)
osnet_x0_5	Market	Duke	29.2 (17.0)
osnet_x0_5	Market	Market	92.8 (81.3)
osnet_x0_25	**Market**	**Duke**	**28.2 (15.1)**
osnet_x0_25	**Market**	**Market**	**91.6 (77.4)**
osnet_ibn_x1_0	**Market**	**Duke**	**47.9 (27.6)**
osnet_ibn_x1_0	**Market**	**Market**	**92.9 (80.0)**
osnet_ain_x1_0	**Market**	**Duke**	**52.4 (30.5)**
osnet_ain_x1_0	**Market**	**Market**	**92.7 (80.3)**

Tabla 4.2: Experimento 1: Resultados de todos los modelos preentrenados en *Market* y testeados en *Duke* y *Market*. Los modelos resaltados en negrita son los que seleccionamos como mejores.

Para terminar, a pesar de que los modelos de *MobileNet* no dan tan buenos resultados, vamos a escoger el *mobilenetx*2_*x*1_0 para poder evaluar con un

Modelo	Train	Test	Rank1 (mAP)
mobilenetv2_x1_0	**Duke**	**Market**	**39.9 (17.0)**
mobilenetv2_x1_0	**Duke**	**Duke**	**74.9 (56.6)**
mobilenetv2_x1_4	Duke	Market	38.0 (16.3)
mobilenetv2_x1_4	Duke	Duke	76.6 (57.6)
osnet_x1_0	Duke	Market	50.2 (22.5)
osnet_x1_0	Duke	Duke	87.0 (72.7)
osnet_x0_75	Duke	Market	47.8 (21.1)
osnet_x0_75	Duke	Duke	85.5 (72.1)
osnet_x0_5	Duke	Market	46.7 (20.1)
osnet_x0_5	Duke	Duke	85.4 (70.7)
osnet_x0_25	**Duke**	**Market**	**41.9 (17.9)**
osnet_x0_25	**Duke**	**Duke**	**81.6 (65.1)**
osnet_ibn_x1_0	**Duke**	**Market**	**57.8 (27.4)**
osnet_ibn_x1_0	**Duke**	**Duke**	**84.5 (69.2)**
osnet_ain_x1_0	**Duke**	**Market**	**61.0 (30.6)**
osnet_ain_x1_0	**Duke**	**Duke**	**85.1 (70.4)**

Tabla 4.3: Experimento 1: Resultados de los modelos preeentrenados en *Duke* y testeados en *Market* y *Duke*. Los modelos resaltados en negrita son los que seleccionamos como mejores.

modelo de diferente método. El motivo por el que cogemos *mobilenetx*2_*x*1_0 y no *mobilenetx*2_*x*1_4 es porque tiene un tamaño de parámetros menor y es más rápido, ya que si nos fijamos en los resultados de la evaluación son muy parecidos. En resumen, seleccionamos los modelos: *mobilenetx*2_*x*1_0, *osnet_x*0_25, *osnet_ibn_x*1_0 y *osnet_ain_x*1_0.

4.3. Experimento 2: Evaluación de modelos con la segmentación de instancias de las imágenes del conjunto de datos

Objetivo: Queremos comprobar el funcionamiento de los modelos utilizando nuestra variación implementada: la segmentación de instancias de las imágenes de los conjuntos de datos como paso inicial.

Descripción: En este experimento compararemos los resultados de los mejores modelos preentrenados testeados con el conjunto de datos originales (modelos resaltados en las Tablas 4.2 y 4.3), con los resultantes de la segmentación de instancias de los conjuntos de datos. El proceso de evaluación es igual que en el experimento anterior. Con los modelos preentrenados testeamos en los diferentes conjuntos de datos, pero esta vez el conjunto de datos será con la segmentación de instancias de las imágenes.

Resultados: En la Tabla 4.4 podemos ver los resultados de los modelos preentrenados con *Market* y testeado con segmentación de instancias de las imágenes

de *Duke*, y en la Tabla 4.5 los resultados obtenidos con los modelos preentrenados en *Duke* y testeado con la segmentación de instancias de las imágenes de *Market*. Los resultados obtenidos son similares a los obtenidos sin hacer la segmentación de instancias.

Modelo	**Train**	**Test**	**Rank1 (mAP)**
mobilenetv2_x1_0	Market	Duke	23.7 (11.8)
mobilenetv2_x1_0	Market	Market	67.7 (45.4)
osnet_x0_25	Market	Duke	25.6 (12.3)
osnet_x0_25	Market	Market	60.6 (35.7)
osnet_ibn_x1_0	Market	Duke	43.5 (23.6)
osnet_ibn_x1_0	Market	Market	75.4 (52.0)
osnet_ain_x1_0	Market	Duke	44.7 (23.3)
osnet_ain_x1_0	Market	Market	75.4 (52.8)

Tabla 4.4: Experimento 2: Resultados de modelos preentrenados en *Market* y testeados en *Duke* y *Market* con la segmentación de instancias de las imágenes de la base de datos.

Modelo	**Train**	**Test**	**Rank1 (mAP)**
mobilenetv2_x1_0	Duke	Market	29.0 (11.8)
mobilenetv2_x1_0	Duke	Duke	46.8 (28.1)
osnet_x0_25	Duke	Market	28.4 (10.7)
osnet_x0_25	Duke	Duke	41.4 (23.9)
osnet_ibn_x1_0	Duke	Market	35.3 (14.3)
osnet_ibn_x1_0	Duke	Duke	54.7 (32.7)
osnet_ain_x1_0	Duke	Market	38.5 (16.2)
osnet_ain_x1_0	Duke	Duke	54.6 (33.2)

Tabla 4.5: Experimento 2: Resultados de modelos preeentrenados en *Duke* y testeados en *Market* y *Duke* con la segmentación de instancias de las imágenes de la base de datos.

Sin embargo, si nos fijamos en algunos ejemplos visuales concretos de la evaluación (Figura 4.4) se ve como al hacer segmentación de instancias obtenemos mejoras. Al sólo aparecer la persona, extrae mejor las características y hace que identifique a la persona sin tener influencia de lo que le rodea. Esto se puede entender mejor en el ejemplo de la Figura 4.5 donde vemos que con el conjunto de datos original la red se 'distrae' con el coche naranja a la hora de detectar a la persona. En cambio, a pesar de no re-identificar del todo bien, con la imagen segmentada el coche ya no existe y sólo se fija en la persona que es lo único que ve y realmente nos interesa, buscando así la similitud sólo referido a la persona.

En el apéndice B podemos ver más resultados donde utilizar como entrada al sistema las imágenes después de aplicarles la segmentación de instancias produce mejores resultados frente al conjunto de datos original.

(a) Base de datos original

(b) Base de datos segmentada

Figura 4.4: Experimento 2: Mejora de testear utilizando la segmentación de instancias de imágenes frente a utilizar el conjunto de datos original. En cada fila, la imagen de la izquierda es la imagen de *consulta*, y el resto son las imágenes más similares encontradas (de izquierda a derecha) por el sistema. El cuadro verde indica acierto mientras que el rojo indica error. Este ejemplo es resultado de evaluar $mobilenetv2\_x1\_0$ preentrenado en $Market$ y testeado en $Duke$.

4.4. Experimento 3: Evaluación de modelos con entrenamientos propios

Objetivo: La finalidad de este experimento es evaluar el funcionamiento de diferentes modelos entrenados con los conjuntos de datos segmentados, intentado así, conseguir una mejor generalización del modelo.

Descripción: Para realizar una comparación justa, se van a realizar dos ensayos, primero el entrenamiento y testeo con el conjunto de datos originales y luego se hará con la segmentación de instancias de las imágenes del conjunto de datos. En este experimento vamos a utilizar dos modelos. Los escogidos son $osnet\_x0\_25$ porque es que tiene menor parámetros y el más rápido (ver Tabla de características 4.1), y $osnet\_ain\_x1\_0$ porque es el que mejores resultados proporciona (Tabla 4.3 y Tabla 4.2). Para la realización del entrenamiento de todos los modelos se han seleccionado 50 épocas, la función de pérdida Softmax y el optimizador Adam con el método de velocidad de aprendizaje 'single_step'. Los resultados del conjunto de datos originales y con la segmentación de instancias de imágenes de $Duke$ y testeados en el conjunto de datos de $Market$ se ven en la tabla 4.6. Los obtenidos entrenando en $Market$ y testeando en $Duke$ se ven en la tabla 4.7.

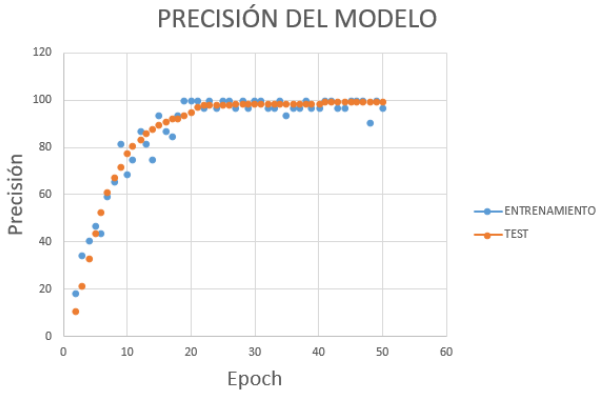
Resultados: Analizando la Tabla 4.6, podemos ver que los modelos resultantes siguen dos tendencias. En primer lugar, los modelos entrenados y testeados en el mismo conjunto de datos obtienen resultados ligeramente inferiores en el caso de utilizar imágenes segmentadas, debido probablemente, a que la red en ciertos casos se apoya en información proveniente del fondo de la imagen. Sin embargo, en el caso de evaluaciones cruzadas, se observa una segunda tendencia de mejora de hasta 4.9 puntos cuando el entrenamiento se ha realizado con imágenes segmentadas frente a un entrenamiento con los datos originales. Por lo tanto, estos resultados muestran como el paso propuesto, de segmentar en detalle las imágenes de las personas, merece la pena de cara a obtener mejor generalización de los modelos.

(a) Base de datos original

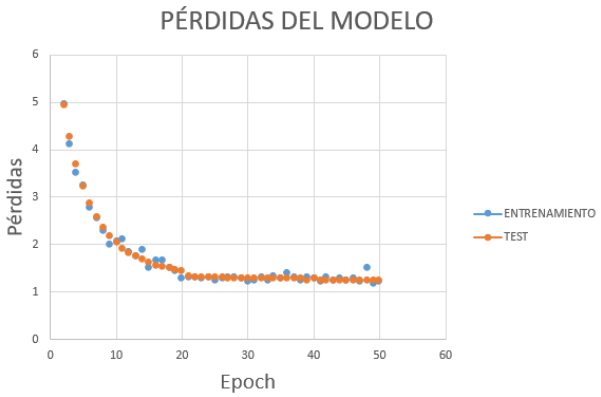
(b) Base de datos segmentada

Figura 4.5: Experimento 2: Ejemplo de identificación con y sin la segmentación propuesta. En cada fila, la imagen de la izquierda es la imagen de *consulta*, y el resto son las imágenes más similares encontradas (de izquierda a derecha) por el sistema. El rojo indica error.Sin embargo, en este ejemplo se muestra la distracción del modelo con el coche naranja si utilizamos el conjunto de datos original, a diferencia de los resultados utilizando segmentación de instancias en los que sólo se fija en la persona. Es un resultado de la evaluación *mobilenetv2_x1_0* preentrenado en *Market* y testeados en *Duke*.

En la Figura 4.6, se puede ver la evolución del entrenamiento del modelo *osnet_x0_25*. Además, en el apéndice C se pueden ver las gráficas restantes de pérdida y precisión de los modelos entrenados desde cero durante 50 épocas.



(a) Precisión del modelo



(b) Pérdida del modelo

Figura 4.6: Experimento 3: Gráficas de precisión y pérdidas en el modelo osnet_x0_25 entrenado con la segmentación de instancias de imágenes del *Market* y testeado con la segmentación de instancias de imágenes de *Duke*. Observamos como el sistema mejora cuando se incrementan las épocas (*epoch*) obteniendo mejores resultados a mayor épocas entrenando.

Modelo	**Train**	**Test**	**Rank1**	**Rank5**	**Rank10**	**mAP**
Modelo entrenado con **segmentación de instancias de imágenes**						
osnet_x0_25	Duke	Market	30.3	49.4	58.7	12.2
osnet_x0_25	Duke	Duke	45.6	62.8	69.6	28.3
osnet_ain_x1_0	Duke	Market	49.0	68.7	75.8	23.4
osnet_ain_x1_0	Duke	Duke	65.0	78.7	82.4	43.2
Modelo entrenado con **conjunto de datos originales**						
osnet_x0_25	Duke	Market	26.2	42.9	51.1	10.3
osnet_x0_25	Duke	Duke	45.6	65.7	73.6	29.5
osnet_ain_x1_0	Duke	Market	44.1	63.8	71.6	19.4
osnet_ain_x1_0	Duke	Duke	68.9	80.7	85.0	46.3

Tabla 4.6: Experimento 3: Comparación de resultados de modelos entrenados por nosotros en el conjunto de datos *Duke* y testeado en *Market* y *Duke* con la segmentación de instancias de imágenes del conjunto de datos frente a los conjunto de datos original.

Modelo	**Train**	**Test**	**Rank1**	**Rank5**	**Rank10**	**mAP**
Modelo entrenado con **segmentación de instancias de imágenes**						
osnet_x0_25	Market	Duke	20.3	33.7	40.3	10.6
osnet_x0_25	Market	Market	55.9	76.8	83.8	33.2
osnet_ain_x1_0	Market	Duke	42.1	56.6	62.8	23.3
osnet_ain_x1_0	Market	Market	77.3	89.7	92.9	53.8
Modelo entrenado con **conjunto de datos originales**						
osnet_x0_25	Market	Duke	8.4	14.8	19.0	3.5
osnet_x0_25	Market	Market	54.6	77.3	85.4	32.9
osnet_ain_x1_0	Market	Duke	26.7	40.8	46.8	13.2
osnet_ain_x1_0	Market	Market				

Tabla 4.7: Experimento 3: Comparación de resultados de modelos entrenados por nosotros en el conjunto de datos *Market* y testeado en *Duke* y *Market* con la segmentación de instancias de imágenes del conjunto de datos frente a los conjunto de datos original.

Capítulo 5

Conclusiones y Trabajo Futuro

5.1. Conclusiones Técnicas

Se ha realizado un estudio de los métodos existentes de re-identificación basados en técnicas de aprendizaje profundo (Deep Learning), centrándonos en redes neuronales convolucionales (CNN). Con este estudio se han verificado y replicado los resultados principales que se encuentran en la literatura más relacionada, realizando las distintas evaluaciones de los métodos de re-identificación, comprobando que los resultados proporcionados son correctos y verificando los problemas abiertos que presentan, en particular, la falta de generalización a entornos diferentes al utilizado en el entrenamiento de los sistemas.

Además, se ha propuesto una variación a los métodos estudiados. Esta variación consiste en realizar un paso de pre-procesado encargado de hacer la segmentación de instancias (en este caso personas) de las imágenes del conjunto de datos, buscando conseguir una mejora del sistema.

Para el estudio y para comprobar el funcionamiento del sistema propuesto hemos realizado tres bloques de experimentos donde se evalúan los distintos modelos preentrenados (experimento 1 y 2) y entrenándolos desde cero (experimento 3) con los distintos conjuntos de datos. La conclusión principal extraída de estos experimentos es que la variación propuesta produce mejora frente a los modelos con el conjunto de datos originales, obteniendo así un sistema más generalizable que es lo que buscábamos. En especial, esta mejora se refleja cuando hacemos nuestros propios entrenamientos desde cero con la segmentación de instancias de las imágenes de los conjuntos de datos. Aún así, cuando usamos los modelos previamente entrenados y testeamos en un conjunto de datos con la segmentación de instancias hecha también se puede ver visualmente en las imágenes de los resultados cómo busca mejor a la persona a re-identificar, ya que con el conjunto de datos original se distrae con objetos de su alrededor.

Con todos estos resultados podemos concluir que merece la pena invertir tiempo en hacer la segmentación de instancias de imágenes como paso inicial del sistema.

Problemas encontrados El principal problema práctico en la realización de este trabajo han sido los requisitos hardware. Para poder escribir código Python instalé Anaconda pero paquetes los Opencv y dlib daban fallo de instalación. Este problema se ha solucionado utilizando el entorno de Google Colab.

La mayoría de los experimentos se han ejecutado en el entorno de Google Colab [1] Este es un entorno gratuito en la nube, que da acceso a la utilización de GPUs, pero que implica una serie de problemas a la hora de lanzar las distintas evaluaciones, ya que el tiempo de uso de la GPU está limitado. Esto ha supuesto que se ralentice el proceso de evaluación y entrenamiento de los modelos junto con la limitación del tiempo de entrenamiento de estos.

5.2. Conclusiones Personales

Gracias a este proyecto he descubierto más en profundidad la visión por computador, en particular temas relacionados con el deep learning, ya que a penas había trabajado en ello.

También me ha servido para ver el funcionamiento del grupo de investigación, trabajando en temas de investigación y aprendiendo las distintas implementaciones que hacían.

5.3. Trabajo Futuro

Se han planteado una serie de ideas que podrían mejorar el sistema propuesto en futuras ampliaciones o mejoras:

- Para seleccionar a la persona que queremos identificar entre las personas de la base de datos podríamos hacer una votación entre las 10 primeras imágenes, donde la persona que tenga más votos será la elegida como más similar. Con esto podríamos hacer el sistema más robusto ya que hay muchas veces que no siempre la persona con menor distancia es la misma que la de la imagen.
- Para conseguir mejores identificaciones podríamos fijarnos y seleccionar primero personas que tengan las regiones locales comunes (por ejemplo el mismo color de sudadera, o de pantalones) a la persona que re-identificaremos para acortar el rango, y luego calcular las distancias métricas de cada una.

[1] `https://colab.research.google.com/notebooks/intro.ipynb`

Apéndice A

Detalles adicionales de los experimentos

A.1. Resultados de todos los modelos preentrenados

Estos resultados se pueden ver en la Tabla A.1.

A.2. Resultado de los dos modelos seleccionados entrenados desde cero durante 50 épocas

Estos dos modelos (*osnet_x*0_25 y *osnet_ain_x*1_0) han sido entrenados y testeados con los conjuntos de datos originales y con la segmentación de instancias de las imágenes de los conjuntos de datos. En la Tabla A.2 podemos ver los resultados de cada entrenamiento con su respectivo conjunto de datos. Hay que recalcar que cuando se dice Modelo entrenado con segmentación de instancias de imágenes, en el testeo también se usan los datos con segmentación de instancias de la base de datos correspondiente. Cuando entrenamos con el conjunto de datos originales es lo mismo, se testea luego con el conjunto de datos originales correspondiente.

A.3. Resultados adicionales

Se han realizado evaluaciones de los diferentes modelos preentrenados en un conjunto de datos adicional. Este conjunto de datos es $MSMT17$ donde sus imágenes fueron realizadas mediante 15 cámaras. Está formado por:

- 4101 identidades de personas.
- 126441 imágenes en total subdivididas en: 32621 de entrenamiento, 11659 de consulta y 82161 de galería.

Los resultados de estas evaluaciones se ven en las Tablas A.3 y A.4 y son con los conjuntos de datos originales.

Modelo	Train	Test	Rank1 (mAP)
mobilenetv2_x1_0	**Market**	**Duke**	**25.1 (13.3)**
mobilenetv2_x1_0	**Market**	**Market**	**86.2 (68.6)**
mobilenetv2_x1_4	Market	Duke	23.1 (12.7)
mobilenetv2_x1_4	Market	Market	86.8 (69.9)
osnet_x1_0	Market	Duke	34.6 (19.7)
osnet_x1_0	Market	Market	94.2 (83.6)
osnet_x0_75	Market	Duke	33.5 (18.8)
osnet_x0_75	Market	Market	93.9 (82.4)
osnet_x0_5	Market	Duke	29.2 (17.0)
osnet_x0_5	Market	Market	92.8 (81.3)
osnet_x0_25	**Market**	**Duke**	**28.2 (15.1)**
osnet_x0_25	**Market**	**Market**	**91.6 (77.4)**
osnet_ibn_x1_0	**Market**	**Duke**	**47.9 (27.6)**
osnet_ibn_x1_0	**Market**	**Market**	**92.9 (80.0)**
osnet_ain_x1_0	**Market**	**Duke**	**52.4 (30.5)**
osnet_ain_x1_0	**Market**	**Market**	**92.7 (80.3)**
mobilenetv2_x1_0	**Duke**	**Market**	**39.9 (17.0)**
mobilenetv2_x1_0	**Duke**	**Duke**	**74.9 (56.6)**
mobilenetv2_x1_4	Duke	Market	38.0 (16.3)
mobilenetv2_x1_4	Duke	Duke	76.6 (57.6)
osnet_x1_0	Duke	Market	50.2 (22.5)
osnet_x1_0	Duke	Duke	87.0 (72.7)
osnet_x0_75	Duke	Market	47.8 (21.1)
osnet_x0_75	Duke	Duke	85.5 (72.1)
osnet_x0_5	Duke	Market	46.7 (20.1)
osnet_x0_5	Duke	Duke	85.4 (70.7)
osnet_x0_25	**Duke**	**Market**	**41.9 (17.9)**
osnet_x0_25	**Duke**	**Duke**	**81.6 (65.1)**
osnet_ibn_x1_0	**Duke**	**Market**	**57.8 (27.4)**
osnet_ibn_x1_0	**Duke**	**Duke**	**84.5 (69.2)**
osnet_ain_x1_0	**Duke**	**Market**	**61.0 (30.6)**
osnet_ain_x1_0	**Duke**	**Duke**	**85.1 (70.4)**

Tabla A.1: Experimento 1: Resultados de todos los modelos preentrenados. Resaltados los modelos que seccionamos como mejores.

Modelo	**Train**	**Test**	**Rank1**	**Rank5**	**Rank10**	**mAP**
Modelo entrenado con **segmentación de instancias de imágenes**						
osnet_x0_25	Duke	Market	30.3	49.4	58.7	12.2
osnet_x0_25	Duke	Duke	45.6	62.8	69.6	28.3
osnet_ain_x1_0	Duke	Market	49.0	68.7	75.8	23.4
osnet_ain_x1_0	Duke	Duke	65.0	78.7	82.4	43.2
osnet_x0_25	Market	Duke	20.3	33.7	40.3	10.6
osnet_x0_25	Market	Market	55.9	76.8	83.8	33.2
osnet_ain_x1_0	Market	Duke	42.1	56.6	62.8	23.3
osnet_ain_x1_0	Market	Market	77.3	89.7	92.9	53.8
Modelo entrenado con **conjunto de datos originales**						
osnet_x0_25	Duke	Market	26.2	42.9	51.1	10.3
osnet_x0_25	Duke	Duke	45.6	65.7	73.6	29.5
osnet_ain_x1_0	Duke	Market	44.1	63.8	71.6	19.4
osnet_ain_x1_0	Duke	Duke	68.9	80.7	85.0	46.3
osnet_x0_25	Market	Duke	8.4	14.8	19.0	3.5
osnet_x0_25	Market	Market	54.6	77.3	85.4	32.9
osnet_ain_x1_0	Market	Duke	26.7	40.8	46.8	13.2
osnet_ain_x1_0	Market	Market	78.3	91.2	94.3	54.1

Tabla A.2: Resultados totales de entrenar los modelos desde cero con la segmentación de instancias de imágenes del conjunto de datos frente al conjunto de datos original

Modelo	**Train**	**Test**	**Rank1 (mAP)**
mobilenetv2_x1_0	MSMT17	Market	37.9 (17.2)
mobilenetv2_x1_0	MSMT17	MSMT17	*
mobilenetv2_x1_4	MSMT17	Market	38.2 (17.4)
mobilenetv2_x1_4	MSMT17	MSMT17	*
osnet_x1_0	MSMT17	Market	66.6 (37.5)
osnet_x1_0	MSMT17	MSMT17	*
osnet_x0_75	MSMT17	Market	63.6 (35.5)
osnet_x0_75	MSMT17	MSMT17	*
osnet_x0_5	MSMT17	Market	64.3 (34.9)
osnet_x0_5	MSMT17	MSMT17	*
osnet_x0_25	MSMT17	Market	59.6 (33.6)
osnet_x0_25	MSMT17	MSMT17	*
osnet_ibn_x1_0	MSMT17	Market	66.7 (39.8)
osnet_ibn_x1_0	MSMT17	MSMT17	*
osnet_ain_x1_0	MSMT17	Market	70.1 (43.3)
osnet_ain_x1_0	MSMT17	MSMT17	*

Tabla A.3: Tabla de resultados de modelos preentrenados en $MSMT17$ y testeados en $Market$. Los resultados con * no se han podido realizar debido al gran tamaño del conjunto de datos $MSMT17$

Modelo	**Train**	**Test**	**Rank1 (mAP)**
mobilenetv2_x1_0	MSMT17	Duke	42.7 (24.1)
mobilenetv2_x1_0	MSMT17	MSMT17	*
mobilenetv2_x1_4	MSMT17	Duke	42.9 (25.3)
mobilenetv2_x1_4	MSMT17	MSMT17	*
osnet_x0_25	MSMT17	Duke	61.9 (42.2)
osnet_x0_25	MSMT17	MSMT17	*
osnet_ibn_x1_0	MSMT17	Duke	68.1 (48.8)
osnet_ibn_x1_0	MSMT17	MSMT17	*
osnet_ain_x1_0	MSMT17	Duke	71.1 (52.7)
osnet_ain_x1_0	MSMT17	MSMT17	*

Tabla A.4: Tabla de resultados de modelos preentrenados en $MSMT17$ y testeados en *Duke*. Los resultados con * no se han podido realizar debido al gran tamaño del conjunto de datos $MSMT17$

Modelo	**Train**	**Test**	**Rank1(mAP)** 10 épocas	**Rank1(mAP)** 20 épocas
mobilenetv2_x1_0	Market + Duke	Duke	13.3 (6.5)	14.9 (7.9)
osnet_x0_25	Market + Duke	Duke	9.1 (4.1)	12.5 (6.2)
osnet_ibn_x1_0	Market + Duke	Duke	25.0 (12.8)	31.5 (17.2)
osnet_ain_x1_0	Market + Duke	Duke	28.5 (14.6)	31.9 (16.6)
mobilenetv2_x1_0	Market + Duke	Market	26.1 (10.9)	31.0 (13.5)
osnet_x0_25	Market + Duke	Market	26.2 (10.2)	29.8 (12.1)
osnet_ibn_x1_0	Market + Duke	Market	42.3 (18.2)	47.8 (21.6)
osnet_ain_x1_0	Market + Duke	Market	43.1 (19.1)	46.9 (21.3)

Tabla A.5: Tabla de resultados de modelos preentrenados en $MSMT17$ y testeados en *Duke*. Los resultados con * no se han podido realizar debido al gran tamaño del conjunto de datos $MSMT17$

También se realizaron entrenamientos con los conjuntos de datos originales durante 10 y 20 épocas, viendo como cuantas más épocas se entrena, mejor resultados se obtiene. Estos entrenamientos se han realizado combinando las imágenes de entrenamiento, galería y consulta. Los resultados se observan en la Tabla A.5

Apéndice B

Mejoras visuales de la segmentación de instancias del conjunto de datos frente al conjunto de datos original

B.1. Modelos preentrenados en *DukeMTMC − reid* y testeados en *Market*1501

Podemos verlo en las Figuras B.1, B.2, B.3 y B.4. En todas estas figuras, en cada fila, la imagen de la izquierda es la imagen de *consulta*, y el resto son las imágenes más similares encontradas (de izquierda a derecha) por el sistema. El cuadro verde indica acierto mientras que rojo indica error.

(a) Base de datos original

(b) Base de datos segmentada

Figura B.1: Ejemplo 1: Modelo osnet_x0_25

(a) Base de datos original

(b) Base de datos segmentada

Figura B.2: Ejemplo 2: Modelo osnet_x0_25

(a) Base de datos original

(b) Base de datos segmentada

Figura B.3: Ejemplo 3: Modelo osnet_ibn_x1_0

B.2. Modelos preentrenados en $Market1501$ y testeados en $DukeMTMC-reid$

Podemos verlo en las Figuras B.5, B.6 y B.7. En todas estas figuras, en cada fila, la imagen de la izquierda es la imagen de *consulta*, y el resto son las imágenes más similares encontradas (de izquierda a derecha) por el sistema. El cuadro verde indica acierto mientras que rojo indica error.

(a) Base de datos original

(b) Base de datos segmentada

Figura B.4: Ejemplo 4: Modelo osnet_ibn_x1_0

(a) Base de datos original

(b) Base de datos segmentada

Figura B.5: Ejemplo 5: Modelo mobilenetv2_x1_0

(a) Base de datos original

(b) Base de datos segmentada

Figura B.6: Ejemplo 6: Modelo mobilenetv2_x1_0

(a) Base de datos original

(b) Base de datos segmentada

Figura B.7: Ejemplo 7: Modelo osnet_x0_25

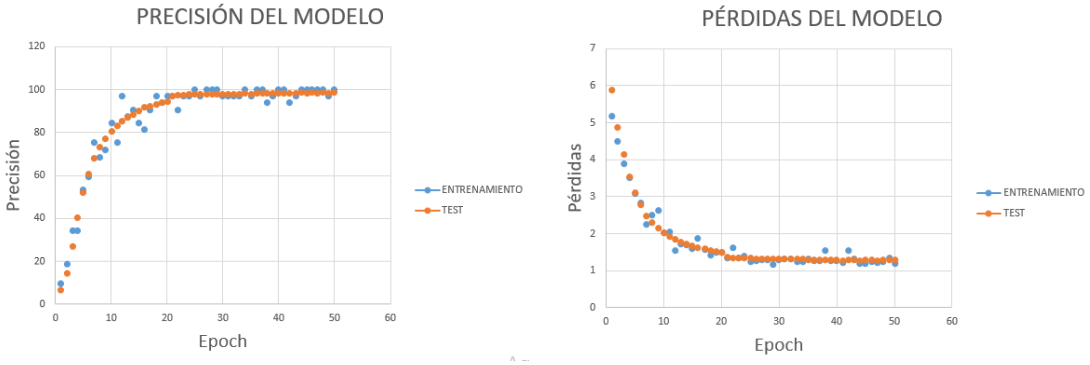
Apéndice C

Comportamiento de modelos durante los entrenamientos

C.1. Entrenamiento con la segmentación de instancias de las imágenes del conjunto de datos de Duke y testeado en Market

C.1.1. OsNet_x0_25

Los gráficos de pérdida y precisión se observan en las Figuras C.1

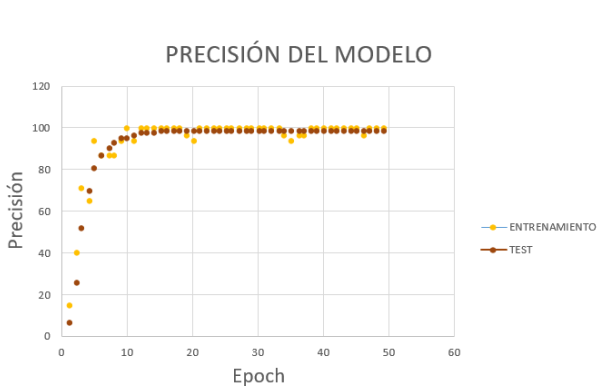


(a) Precisión del modelo (b) Pérdida del modelo

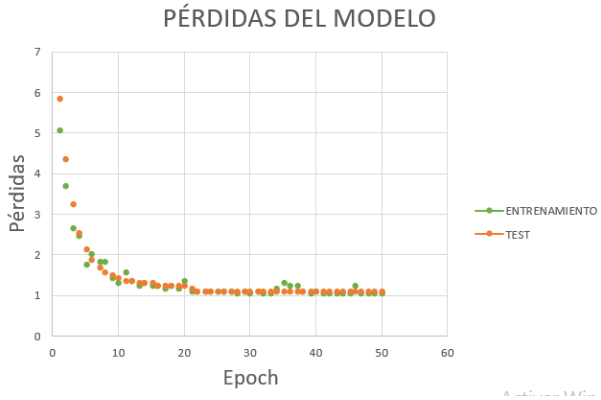
Figura C.1:

C.1.2. osnet_ain_x1_0

Los gráficos de pérdida y precisión se observan en las Figuras C.2



(a) Precisión del modelo



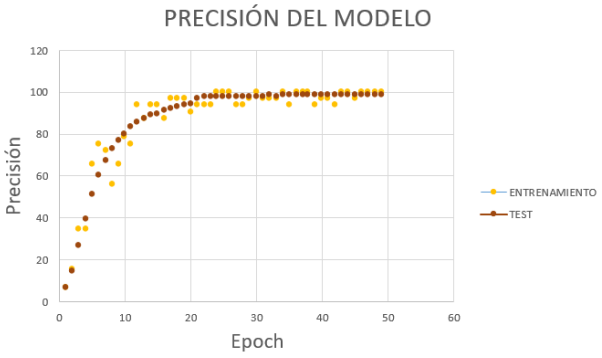
(b) Pérdida del modelo

Figura C.2:

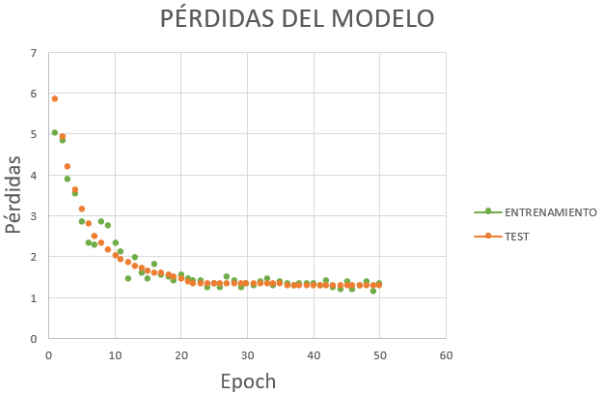
C.2. Entrenamiento con la segmentación de instancias de las imágenes del conjunto de datos de Duke y testeado en Duke

C.2.1. OsNet_x0_25

Los gráficos de pérdida y precisión se observan en las Figuras C.3



(a) Precisión del modelo

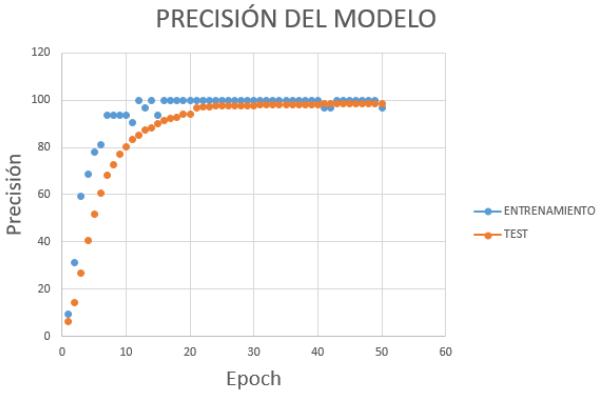


(b) Pérdida del modelo

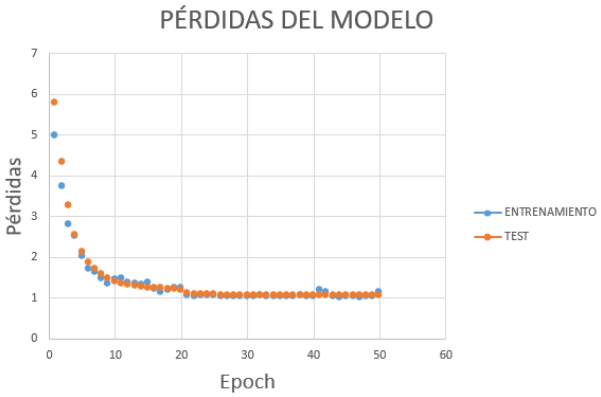
Figura C.3:

C.2.2. osnet_ain_x1_0

Los gráficos de pérdida y precisión se observan en las Figuras C.4



(a) Precisión del modelo



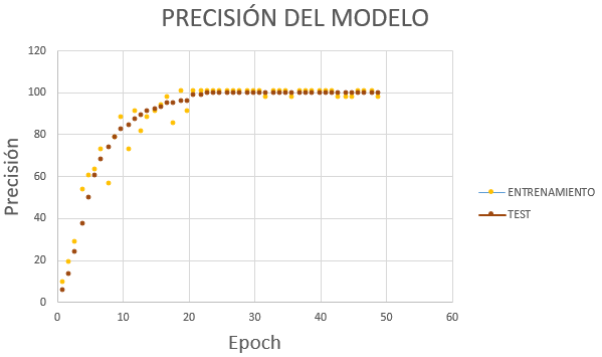
(b) Pérdida del modelo

Figura C.4:

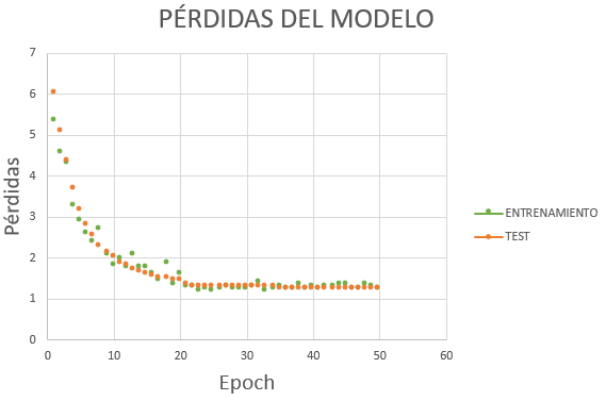
C.3. Entrenamiento con el conjunto de datos original de Duke y testeado en Market

C.3.1. OsNet_x0_25

Los gráficos de pérdida y precisión se observan en las Figuras C.5



(a) Precisión del modelo

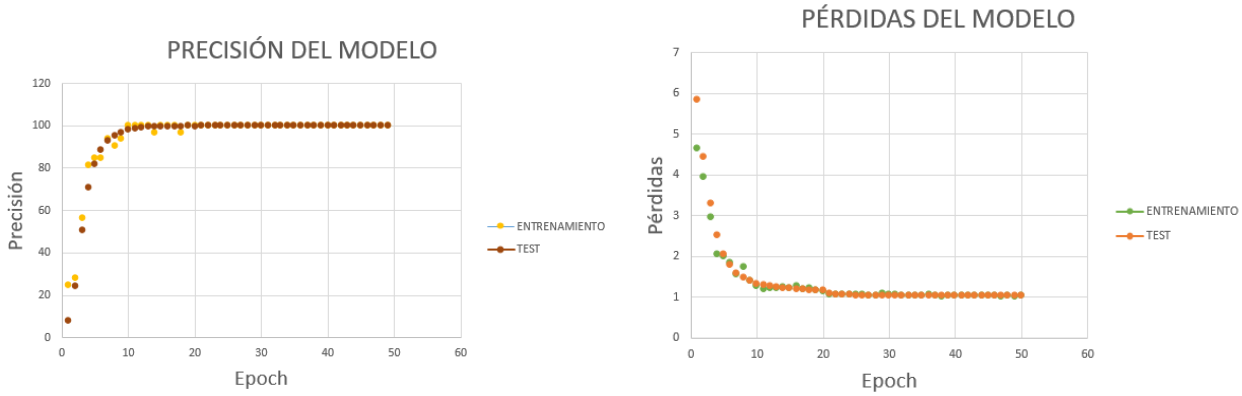


(b) Pérdida del modelo

Figura C.5:

C.3.2. osnet_ain_x1_0

Los gráficos de pérdida y precisión se observan en las Figuras C.6



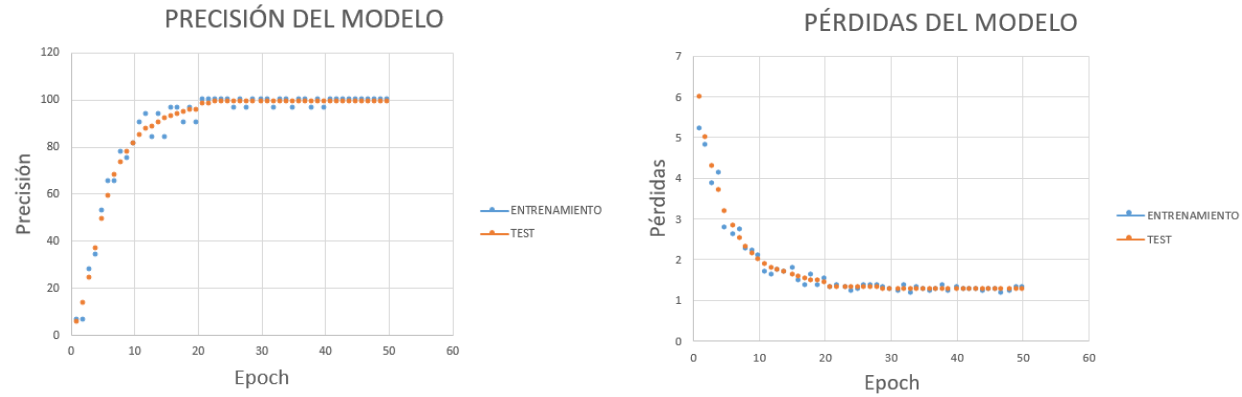
(a) Precisión del modelo (b) Pérdida del modelo

Figura C.6:

C.4. Entrenamiento con conjunto de datos originales de Duke y testeado en Duke

C.4.1. OsNet_x0_25

Los gráficos de pérdida y precisión se observan en las Figuras C.7

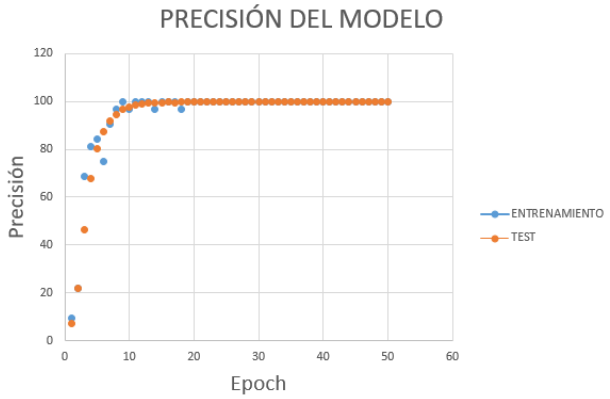


(a) Precisión del modelo (b) Pérdida del modelo

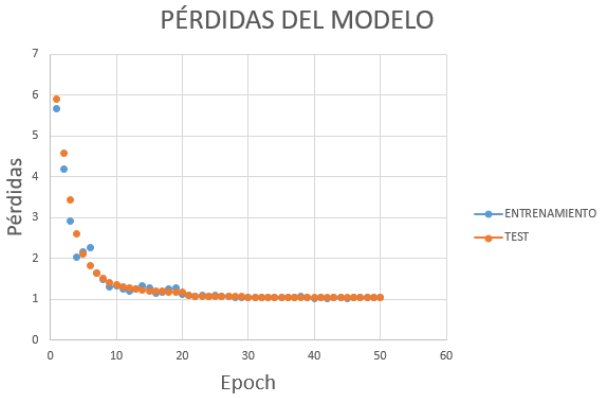
Figura C.7:

C.4.2. osnet_ain_x1_0

Los gráficos de pérdida y precisión se observan en las Figuras C.8



(a) Precisión del modelo



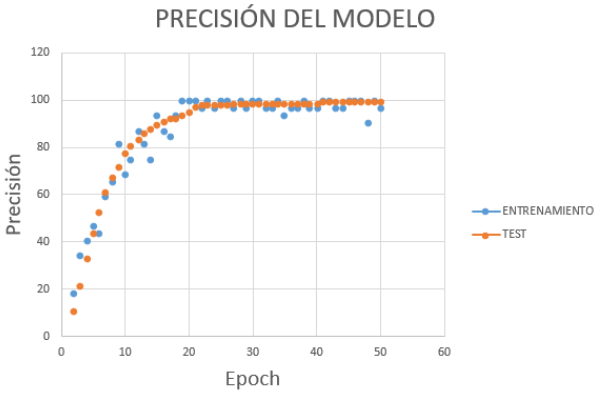
(b) Pérdida del modelo

Figura C.8:

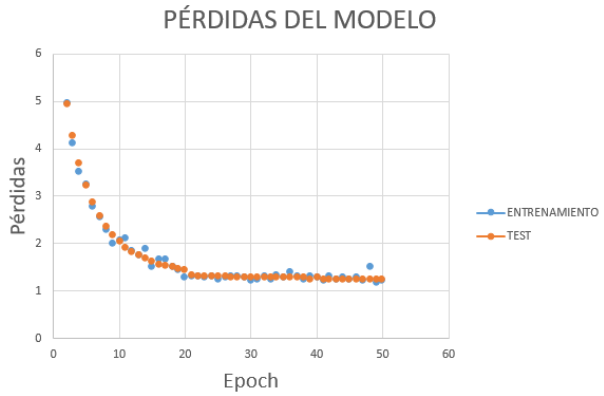
C.5. Entrenamiento con la segmentación de instancias de las imágenes del conjunto de datos Market y testeado en Duke

C.5.1. OsNet_x0_25

Los gráficos de pérdida y precisión se observan en las Figuras C.9



(a) Precisión del modelo

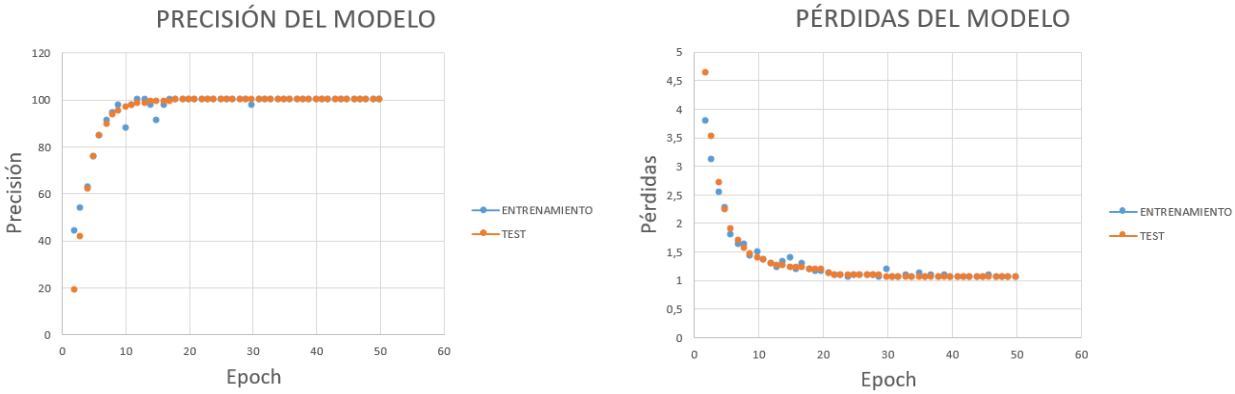


(b) Pérdida del modelo

Figura C.9:

C.5.2. osnet_ain_x1_0

Los gráficos de pérdida y precisión se observan en las Figuras C.10



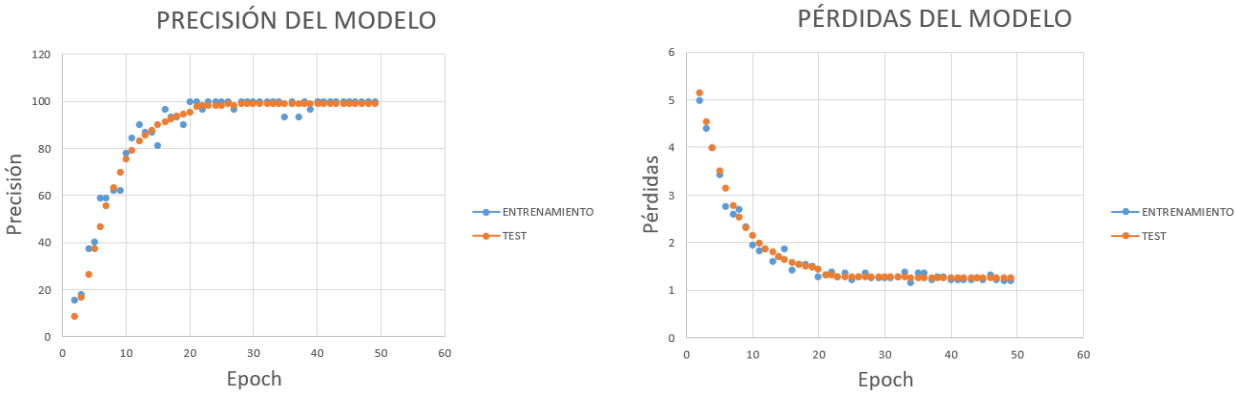
(a) Precisión del modelo (b) Pérdida del modelo

Figura C.10: División de tareas del proyecto

C.6. Entrenamiento con el conjunto de datos original de Market y testeado en Duke

C.6.1. OsNet_x0_25

Los gráficos de pérdida y precisión se observan en las Figuras C.11

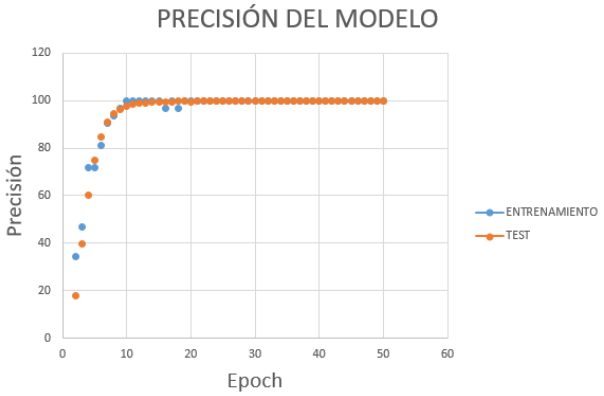


(a) Precisión del modelo (b) Pérdida del modelo

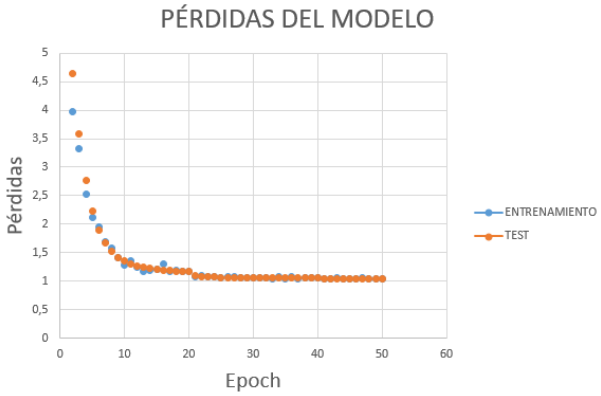
Figura C.11:

C.6.2. osnet_ain_x1_0

Los gráficos de pérdida y precisión se observan en las Figuras C.12



(a) Precisión del modelo



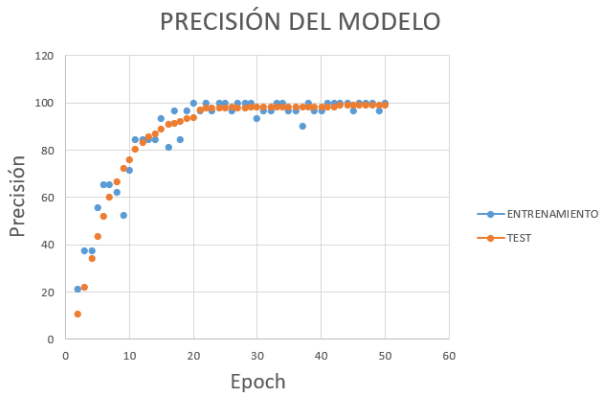
(b) Pérdida del modelo

Figura C.12:

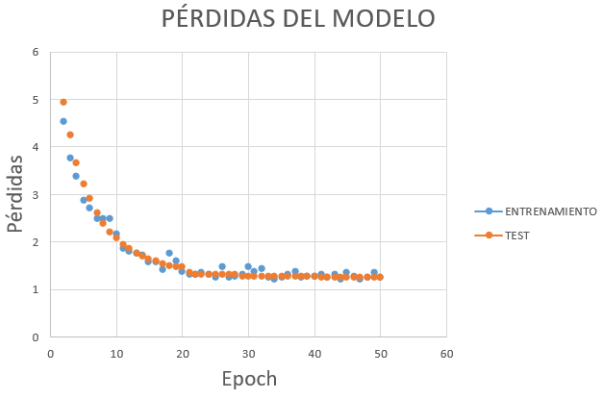
C.7. Entrenamiento con la segmentación de instancias de las imágenes del conjunto de datos Market y testeado en Market

C.7.1. OsNet_x0_25

Los gráficos de pérdida y precisión se observan en las Figuras C.13



(a) Precisión del modelo

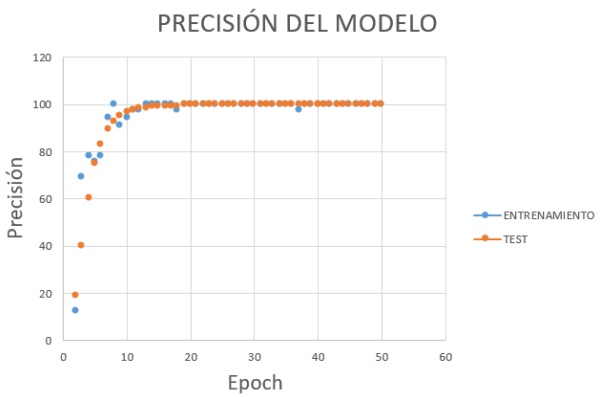


(b) Pérdida del modelo

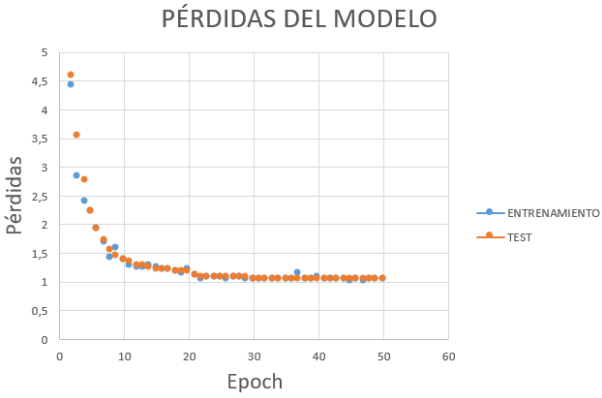
Figura C.13:

C.7.2. osnet_ain_x1_0

Los gráficos de pérdida y precisión se observan en las Figuras C.14



(a) Precisión del modelo



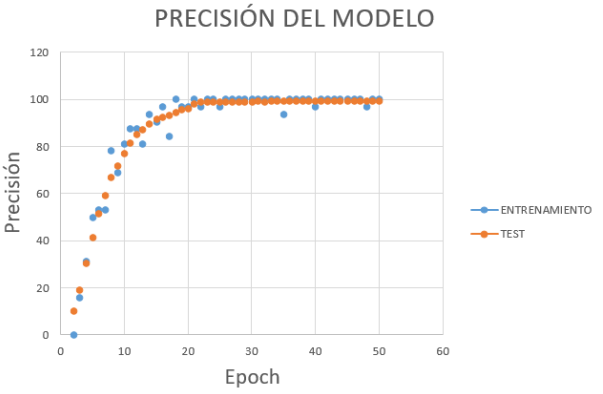
(b) Pérdida del modelo

Figura C.14:

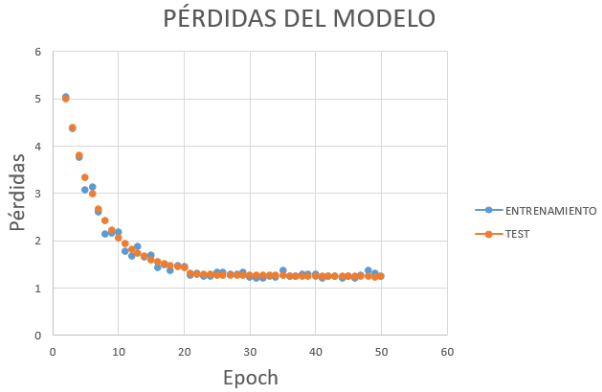
C.8. Entrenamiento con el conjunto de datos original de Market y testeado en Market

C.8.1. OsNet_x0_25

Los gráficos de pérdida y precisión se observan en las Figuras C.15



(a) Precisión del modelo

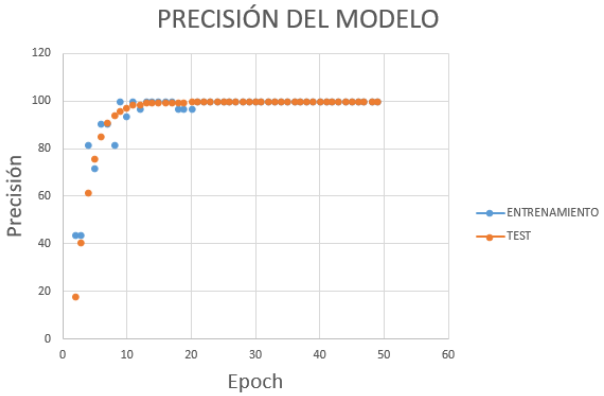


(b) Pérdida del modelo

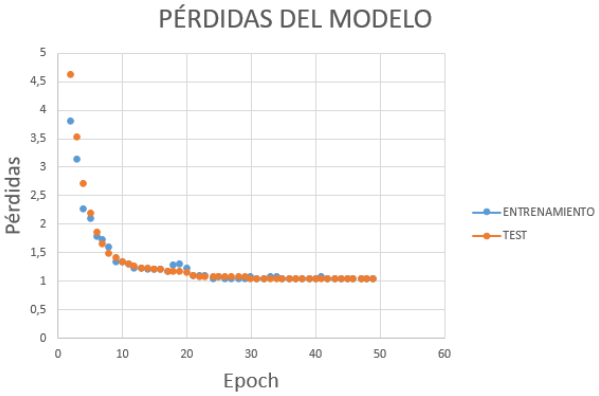
Figura C.15:

C.8.2. osnet_ain_x1_0

Los gráficos de pérdida y precisión se observan en las Figuras C.16



(a) Precisión del modelo



(b) Pérdida del modelo

Figura C.16:

Bibliografía

[1] Guosheng Hu, Yongxin Yang, Dong Yi, Josef Kittler, William Christmas, Stan Z Li, and Timothy Hospedales. When face recognition meets with deep learning: an evaluation of convolutional neural networks for face recognition. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 142–150, 2015.

[2] Sorin Grigorescu, Bogdan Trasnea, Tiberiu Cocias, and Gigel Macesanu. A survey of deep learning techniques for autonomous driving. *Journal of Field Robotics*, 37(3):362–386, 2020.

[3] Cordova Eras and Wilson Steeven. Implementación de un sistema de reconocimiento de distanciamiento social como medida preventiva para covid 19 usando deep learning. B.S. thesis, Machala: Universidad Tëcnica de Machala, 2020.

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

[5] Zetao Chen, Adam Jacobson, Niko Sünderhauf, Ben Upcroft, Lingqiao Liu, Chunhua Shen, Ian Reid, and Michael Milford. Deep learning features at scale for visual place recognition. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3223–3230. IEEE, 2017.

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

[7] Wei Li, Rui Zhao, Tong Xiao, and Xiaogang Wang. Deepreid: Deep filter pairing neural network for person re-identification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 152–159, 2014.

[8] Yifan Sun, Liang Zheng, Yi Yang, Qi Tian, and Shengjin Wang. Beyond part models: Person retrieval with refined part pooling (and a strong convolutional baseline). In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 480–496, 2018.

[9] Peter Harrington. *Machine learning in action*. Manning Publications Co., 2012.

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

[11] Kaiyang Zhou, Yongxin Yang, Andrea Cavallaro, and Tao Xiang. Learning generalisable omni-scale representations for person re-identification. *arXiv preprint arXiv:1910.06827*, 2019.

[12] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[13] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[14] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[15] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE transactions on pattern analysis and machine intelligence*, 39(6):1137–1149, 2016.

[16] Ergys Ristani, Francesco Solera, Roger Zou, Rita Cucchiara, and Carlo Tomasi. Performance measures and a data set for multi-target, multi-camera tracking. In *European Conference on Computer Vision*, pages 17–35. Springer, 2016.

[17] 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.