Real-Time Person Re-Identification for Mobile Robots to Improve Human-Robot Interaction

Per Antoine Carlsen

Thesis submitted for the degree of
Master in Robotics and Intelligent Systems
60 credits

Department of Informatics
Faculty of mathematics and natural sciences

UNIVERSITY OF OSLO

Spring 2019

Real-Time Person Re-Identification for Mobile Robots to Improve Human-Robot Interaction

Per Antoine Carlsen

Abstract

Mobile robots operating in seniors' homes can serve as social companions and assist with daily tasks, thus enhancing the seniors' quality of life [104]. In order for robots to assist seniors, it is crucial that they are equipped with sets of social and interactive skills to enable them to have natural and personalized interactions. Personalized interactions, such as using patients' proper names or remembering personal preferences, is necessary to establish strong social relationships [4, 45], and is a key factor to improve trust in human-robot interaction [37]. A prerequisite for robots to achieve personalized interactions, however, is the ability to automatically recognize and re-identify people around them [4]. Existing person re-identification systems for mobile robots are highly restricted in terms of where robots can operate, and do not stimulate natural and personalized interactions because they need preliminary knowledge about the robot's users [12, 18], rely on facial cues [113, 115], or use data collected from external sensors [45]. This thesis introduces two lightweight Siamese convolutional neural networks, *LuNet Light* and *LuNet Lightest*, designed for the problem of person re-identification in a robotic setting without relying on the aforementioned restrictions. Despite being significantly more lightweight than other person re-identification systems [3, 120], LuNet Lightest achieves near state-of-the-art results on the MARS dataset evaluation protocols [135]. This thesis additionally presents a set of evaluation measures tailored to evaluate re-identification systems for robots operating in various environments. When simulating crowded environments, LuNet Lightest reaches 92.4% balanced accuracy on the proposed evaluation protocol. As a result of the lightweight architecture, LuNet Lightest achieves real-time frame-rates of 71.6 frames per second when using a GPU, 33.9 frames per second when using a CPU without GPU, and 15.7 frames per second when using only one core of the same CPU, rendering the proposed system highly suitable for low-cost, hardware-constrained robots. The proposed person re-identification system will enable assistive mobile robots to robustly and accurately identify their users, and is a preliminary step to improve trust and attain natural and personalized interaction between robots and patients.

Acknowledgements

This research project was conducted over a two-semester exchange period at UC San Diego, USA, through the Collaboration on Intelligent Machines (COINMAC) program. I am very grateful to my supervisor, Professor Jim Tørresen, for having arranged the exchange program and for accepting and supervising this work.

I would like to thank everyone at the Healthcare Robotics Lab at UC San Diego, lead by Professor Laurel Riek, for being so welcoming and for providing equipment and a good working environment. A special thanks goes to PhD students Angelique Taylor and Darren Chan. This project would not have been possible without your endless support, feedback, and our numerous meetings and brainstorming sessions. I am deeply grateful for everything you have taught me.

Finally, I would like to thank my family and especially my parents, Anne Le Droguen and Ulf Carlsen, for supporting me during this year abroad. Thanks to all co-students and friends, both in Oslo and San Diego, for all the good memories these past two years.

Contents

Abbreviations

ANN	Artificial Neural Network.
BB	Bounding Box.
CMC	Cumulative Matching Characteristics.
CNN	Convolutional Neural Network.
CPU	Central Processing Unit.
DL	Deep Learning.
DPM	Deformable Part Model.
FC	Fully Connected.
FN	False Negative.
FNR	False Negative Rate.
FP	False Positive.
FPR	False Positive Rate.
FPS	Frames Per Second.
FTR	False Target Rate.
GMMCP	Globally Optimal Generalized Maximum Multi Clique Problem.
GPU	Graphical Processing Unit.
LReLU	Leaky Rectified Linear Unit.
mAP	mean Average Precision.
MARS	Motion Analysis and Re-identification Set.
MECS	The Multimodal Elderly Care Systems.
ML	Machine Learning.
PCA	Principal Component Analysis.
pID	Person Identity.
re-ID	re-identification.
RFID	Radio Frequency Identification.
RGB	Red, Green, Blue.
RGB-D	Red, Green, Blue, Depth.
RNN	Recurrent Neural Network.
ROC	Receiver Operating Characteristics.
t-SNE	T-distributed Stochastic Neighbor Embedding.
TN	True Negative.
TNR	True Negative Rate.
TP	True Positive.
TPR	True Positive Rate.
TTR	True Target Rate.

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

The healthcare sector is not sustainable for the rapid aging of the world's population both in terms of cost and in terms of the demand of health aid [44, 48, 55]. Nearly 90% of the American population over the age of 65 wish to live in their homes for as long as possible rather than in assisted living communities [98]. These seniors will, however, require assistance, and with the shortage of professional caretakers, this may imply heavy burdens on non-professional caretakers such as family and friends. Furthermore, in many cases these seniors live alone, which can result in loneliness and reduced quality of life [98].

Many research projects are devoted to investigating how robots can reduce the workload applied to both professional and non-professional caretakers [26, 35, 36, 104, 106, 108, 117]. A possible solution is to develop mobile household robots that can be placed in the homes of the elderly patients who wish to live in their own homes for as long as possible. Household robots can for instance assist in simple daily tasks or serve as social companions [104]. This could save a significant amount of valuable time and workload for healthcare workers, as they would not have to commute to the patients' homes as frequently.

The Multimodal Elderly Care Systems (MECS) [104] is a project that investigates how a robot companion can assist seniors that want to stay in their current homes. Companion robots, such as the MECS robot, will be expected to take on social roles, and it will be crucial that they possess adequate social and communication skills. In order for the companion robots to fulfill their social obligations it is essential that the interactions between robots and humans are natural and personalized [4, 43]. To achieve personalized interactions, it is crucial that robots can automatically recognize and identify people around them [4]. Moreover, robots need the ability to recall people whom they have previously interacted with (acquaintances), as well as remember new individuals for possible future interactions.

Enabling companion and household robots with the skill of identifying people will allow them to store personalized information from previous encounters to generate more complex and rich interactions (see Figure 1.1). Patient information such as name, culture, age, personal preferences, and communication patterns will permit robots to have engaging conversations and improve human-robot teamwork in cooperative tasks. For instance, instead of using general greetings, identity recognition can enable robots to call people by their proper names, which is necessary to establish strong social relationships [4, 45]. Furthermore, using personal names during interaction is important because robot personality and adaptability are key factors to improve trust in human-robot interaction (HRI) [37]. Trust is arguably one of the most important factors when it comes to acceptance of having an autonomous robot in your private home, especially amongst the elderly population who often have limited experience with and knowledge about new technology.

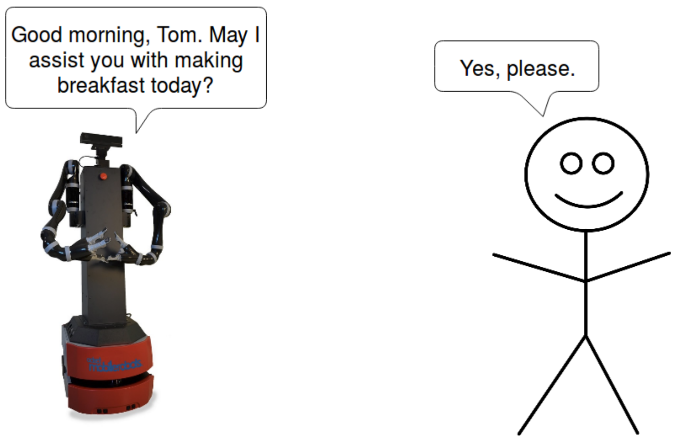

Good morning, Tom. May I assist you with making breakfast today?
Yes, please.

Figure 1.1: Person identification promotes trust, politeness, and good human-robot interaction by enabling robots to use personal names and information acquired from previous interactions. The Mobile robot depicted is from Adept MobileRobots [89].

However, mobile robots are not yet equipped with systems that can automatically and robustly identify people without strictly relying on facial cues [56, 105, 113, 115], information from external sensors [45], or appearance information given in advance [12, 18]. Face recognition and identification requires people to situate themselves in such a way that their faces are clearly visible to the robot, which is often unnatural and inconvenient. Data from external sensors, such as wearable radio-frequency identification (RFID) tags or surveillance cameras, and requiring appearance information in advance highly restricts where and with whom a mobile robot can operate. Furthermore, relying on external sensors mounted in patients' homes can be a severe privacy threat and should be avoided.

1.2 Research Goals

This work aims at developing a person re-identification system specifically tailored for mobile robots. This project investigates whether a person re-identification system achieving state-of-the-art results on public dataset benchmarks can have any practical value for mobile robots.

To that end, the proposed person re-identification system is evaluated using:

1. A public person re-identification dataset benchmarks that is popularly used amongst computer vision (CV) researchers in the field of person re-identification.

2. A new set of evaluation metrics that are developed in this research project. These metrics are specifically designed to evaluate person re-identification in a realistic robotic scenario.

3. Time measurements to investigate whether a top-performing person re-identification system can achieve the high frame rate required by robots that are expected to operate in real-time in dynamic environments.

1.3 Contributions

The contributions of this thesis are threefold:

First, this thesis proves that a shallow and lightweight person re-identification system can obtain results comparable to state-of-the-art systems on one of the largest existing person re-identification benchmarks. This shows that the complex and resource-consuming architectures that most state-of-the-art re-identification systems use are not necessary to achieve high performance. Amongst all top-performing re-identification systems, the proposed architecture is, to our knowledge, the most lightweight person re-identification system.

Second, this thesis investigates common assumptions in the CV field of person re-identification and why these hinder state-of-the-art person re-identification systems from being deployed on mobile robots. To that end, a new set of evaluation metrics that more realistically reflect the typical environment of a mobile robot are presented. While it is shown that this makes the person re-identification problem more challenging, the proposed system achieves up to 92.4% balanced accuracy score on the newly proposed evaluation metrics.

Finally, time measurements in terms of frames per second (FPS) are measured to analyze whether the proposed system is capable of handling the high efficiency requirements even on the low-cost hardware components that are typically found on mobile robots. To evaluate the system efficiency for robots with different hardware, the frame rate is measured using three different settings:

- *GPU*: a 33MHz Intel GeForce GTX is used, which may be representative for high-end mobile robots.
- *CPU*: a 2.60 GHz Intel Core i7 with eight cores and no GPU is used, which may be representative for the majority of mobile robots.
- *single core CPU*: only one of the eight core on the same CPU with no GPU is used, which may be representative for lightweight and low-cost mobile robots.

These definitions will be used throughout this thesis. Full details regarding hardware and software specifications used may be found in the appendix Section 8.1.

The measures show that with a GPU, the system greatly exceeds the real-time frame rate requirement of 30 FPS. With the CPU, the proposed system can process 33.9 FPS, which also fulfills the real-time requirements. With the single core CPU, the proposed system achieves 15.7 FPS. This is, to our knowledge, the first research within the CV field of person re-identification that reports system efficiency in terms of frame rate. The efficiency results show that it is possible to achieve state-of-the-art accuracy on public benchmarks without neglecting the real-time frame-rate requirement of robots.

To summarize, this thesis presents a fully automated person identification framework for a mobile robot. It is shown that robot perception systems can draw inspiration from state-of-the-art CV person re-identification to enhance robot performance. The proposed framework is a foundation for designing robots with personalized behavior and more sophisticated social skills.

1.4 Structure of the Thesis

This thesis is structured in a manner that chronologically provides insight into the topic of person re-identification and common application areas, details about the proposed system, and finally introducing the newly proposed evaluation metrics along with the obtained results. The chapters are structured as follows:

- Chapter 2 describes various robotic applications that take use of person re-identification systems, and additionally provides insight into the CV research field of person re-identification. Various shortcomings, restrictive assumptions and research gaps are identified and discussed.
- Chapter 3 summarizes the public datasets popularly used for person re-identification research, along with discussing aspects considered when selecting an appropriate dataset for this work.

- Chapter 4 looks into the details of the proposed person re-identification pipelines, *LuNet Light* and *LuNet Lightest*.
- Chapter 5 first presents the results obtained on a public person re-identification benchmarks. Then, the newly proposed evaluation metrics are described along with the results obtained. Finally, the system efficiency of the proposed models are discussed.
- Chapter 6 presents the conclusions that can be drawn from this research project.
- Chapter 7 discusses some unexplored aspects and possible topics for future research.

Chapter 2

Background

People re-identification, commonly known as *re-identification* or simply *re-ID*, is an important topic that relates to robotics, HRI, and CV. In robotics, perceiving and re-identifying people is necessary in order for robots to understand their surroundings. In HRI, people re-identification is an essential component in order to give robots the social skills needed for natural and personalized interaction. In CV, the challenging problem of re-identifying people is of high interest due to its significance in many applications including surveillance and robotics.

This chapter is divided into four parts. The first section (Section 2.1) gives an overview regarding the problem of person re-identification in the context of robotics. The second section (Section 2.2) discusses why person re-identification is essential in HRI and how researchers have used re-identification on various mobile robots to improve HRI. The third section (Section 2.3) looks into how person re-identification is approached amongst researchers in the CV community. Finally, Section 2.4 concludes this chapter by summarizing shortcomings of the discussed re-identification systems.

2.1 Overview

Within the context of robotics, re-identification is the task of re-identifying people that leave the robot's field of view and return at a later point in time. This thesis identifies six criteria that need to be met when designing such a system.

1. Robots should not need to be explicitly told to re-identify one or multiple specific people, but should instead perform the re-identification in an automated manner. In other words, the re-identification system needs to operate without requiring preliminary knowledge about the specific individual(s) to be re-identified.

2. The robot needs to keep a database of people whom it has encountered before and may re-encounter during future interactions. This database needs to be able to expand in order to collect information about new people as the robot encounters them.

3. The robot needs to automatically distinguish between acquaintances (previously recognized individuals) and those it has not interacted with before. If the robot recognizes an acquaintance, it needs to perform the re-identification task to associate this person with the correct identity. The robot needs to automatically store memory of people it has not encountered before, so that they can be re-identified the next time they encounter the robot.

4. The robot needs to perform re-identification naturalistically and candidly, without interfering with activities of humans.

5. The re-ID system needs to be as lightweight as possible to meet the real-time constraint of mobile robots. Real-time feedback is crucial if robots are to reach our high expectations and ensure smooth and efficient operation.

6. Finally, the re-identification needs to be robust towards factors that may change over time. Such factors include varying light conditions, changes in human pose and changes in sensor orientation (viewpoint).

2.2 Human-Robot Interaction

Human-Robot Interaction (HRI) is a multidisciplinary field combining theory from robotics, social science, and artificial intelligence, which aims to give robots socially appropriate interaction skills. As robots are transitioning from closed industrial environments to dynamic, human-centered environments, many settings require robots to interact, communicate, and cooperate closely with humans. In the healthcare sector, for example, mobile robots can function as socially assistive coaches for rehabilitative walking and orientation training, where they work closely alongside patients [36]. Natural and socially appropriate interaction between humans and robots is essential to ensure that these human-robot environments are pleasant and safe. HRI focuses on understanding how perception, verbal and nonverbal communication, and emotions affect human interactions and how this knowledge can be used to improve interaction between humans and robots [103].

In order for robots to understand and achieve social interaction, it is necessary that they first perceive people in their environment. Perceiving people is needed in order to navigate efficiently and safely [7], to approach people in an appropriate manner [106] and to initiate and maintain social interaction [103]. In other words, robustly and reliably perceiving people will enhance the interactive skills and overall performance of robots operating in human-centered environments. Furthermore, performance is known to be a great factor when it comes to the trustworthiness of the robot [37], allowing them to more seamlessly fit in to human-centered environments.

In addition to solely perceiving people, robots have to distinguish between acquaintances and strangers [103]. Recognizing people's identities is an essential skill to establish and maintain social relationships between humans and robots [46]. For instance, people re-identification can enable robots to interact and communicate in a polite and appropriate manner by using personal names in greetings. It additionally enables robots to establish long-term relationships with people whom they interact with frequently. This is important for assistive robots as they often have to learn and remember personal attributes and preferences of long-term users. Furthermore, robots can use identity information to keep their attention and resources to specific people of interest.

2.2.1 Person re-ID on Robotic Platforms to Improve HRI

For robots operating in human-centered environments, people perception is typically achieved through an end-to-end system consisting of three steps: people detection, people tracking, and people re-identification. Detection is the task of locating individual people in video, without association to the person's identity. The detected people are represented by bounding boxes (BBs), which are rectangles representing their position and size in image coordinates (an example is shown in Figure 2.3). Tracking uses the BBs and aims at keeping track of people over time [97]. While both tasks of detection and tracking are well-defined, re-identification is often treated differently depending on the robotic application and the task the robot is designed for. Re-identification is most commonly used to recognize the identity of people as they leave and re-enter the robot's environment, which is typically achieved by keeping a database of one or multiple people of interest. However, as robots meet and interact with new people, the re-identification system should also automatically register new identities in order to recognize these people during future interactions.

Although the detection and tracking steps are commonly seen on human-centered mobile robots [29], considerably fewer works include the re-identification step. Since people re-identification on mobile robots is the main topic of this thesis, this section will only consider mobile robots that are implemented with people re-identification as a part of their perception system.

Service and Social Robots

Service robots and social robots are autonomous, and often mobile, robots that can interact socially with people. Service robots are additionally designed to help people by performing various useful tasks [88]. To ensure that the robots interact in a natural manner, it is crucial that they can identify people in their environment [4].

There are several different ways to approach the re-identification problem. Some researchers explore using wearable technology to identify people [4, 45, 86]. Kanda et al. [45] use RFID (radio frequency identification) tags to identify people from a shopping mall information robot. Alonso-Martin et al. [4] use electronic beacons to develop an interactive robot capable of identifying multiple people. Ramirez et al. [86] examine how mobile robots should approach people. They use detectable helmets to identify the target to approach. Using external sensors, including wearable technology, simplifies the re-identification problem, but it is also impractical and highly restricts the operational area of the robot. To overcome these restrictions, robots should preferably only rely upon data gathered by ego-centric sensors (i.e. sensors mounted on the robot itself).

Another common approach is to perform re-identification using face cues. Wang et al. [113] created TritonBot, a tour guide and receptionist robot in a building at UC San Diego. It keeps a database of known faces, and automatically attempts to re-identify acquaintances and register new people it meets. This enables TritonBot to use personalized greetings whenever it meets acquaintances. Similarly, Wang et al. [115] developed a perception system for a service robot that automatically detects faces, extracts discriminative features using convolutional neural networks (CNNs) and register them in a database. Their system uses these features to re-identify previously seen people in addition to automatically registering new people on-the-fly. Other researchers have used face recognition to recognize users on interactive, educational robots that asks science questions [56], and on a mobile security robot [105].

Biometrics, such as face cues, provide good features for both short-term and long-term re-identification. However, face recognition also requires users to directly face the robot at a certain optimal distance. This is undesirable because it heavily restricts how people position and orient themselves in interactive and cooperative human-robot tasks. Wang et al. [113] reported that people facing sideways often resulted in misclassification, confirming that it is desirable to avoid the heavy positional and orientational restrictions.

Belletto and Hu [11, 12] combine biometric face features with person height and a color histogram of the human torso to perform person re-identification from a mobile service robot. The authors observe that during close interaction, the upper part of the human torso is the only body-part visible to their robot. A color histogram of the torso is therefore combined with person height and face features to create an appearance model used for re-identification. They match the detected people against a small, pre-recorded database of 13 subjects, and the robot cannot create appearance models of new people on the fly. Furthermore, they allegedly evaluate the system on video recorded in a office space with good light conditions and people wearing very distinguishable colors on their upper-body, favoring the color-based feature extraction approach. Zhang et al. [130] create a similar system that additionally can register new people. They do however only rely on color features and are more concerned about addressing difficulties caused by short occlusions than person re-identification over a longer time span.

An et al. [6] propose a person re-identification and action recognition algorithm for a mobile service robot. They argue that it is impractical for service robots to require a registration phase in order to identify people. By combining 3D body part information with color histograms, the authors propose a re-identification algorithm that learns on the fly without the need of an enrollment phase or pre-recorded datasets. Like the majority of the research methods discussed in this section, their system is only capable of identifying one "person under service", and can therefore not identify multiple people at once.

Cosar et al. [19] combine the person re-identification module of Li et al. [65] with calculation of dimension of various body parts to create a person re-identification system for mobile robots. They do however train their model on videos of the very same individuals used for evaluation, meaning that their system is unsuitable for robots that continuously meet new people.

2.2.2 Person-Following Robots

Robots that can follow people are highly useful in several situations. They can be used as personal assistants in rehabilitation [26, 35, 36, 106, 108, 117], as automatic push-carts in shopping malls [81], and for various tasks in the service sector [5, 14, 50, 53, 54, 60, 91]. A system that identifies people is essential for all of these robots because they have to distinguish between the person they follow and other people in their environment. Figure 2.1 depicts a challenging situation for a person-following robot due to highly similar person appearances.

The ROREAS project [26, 35, 36, 117] and the ROGER project [106, 108] both develop rehabilitation robots with person-following behavior to accompany patients in walking exercises. These robots have systems to identify the person to follow. They do however require an "enrollment

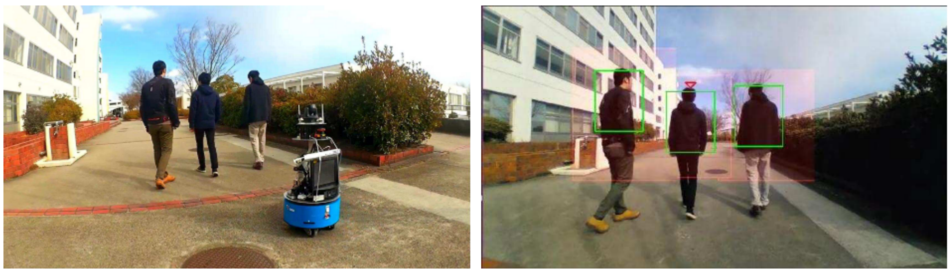


Figure 2.1: Left: a person-following robot. Right: the robot's field of view with the target to follow in the middle. Similar appearances due to clothing and hair color can make it challenging to distinguish between the target to follow and other people. Image from [53].

phase" (see Figure 2.2) where a person appearance model is built beforehand, and can thus only identify one specific individual, which is substantially easier than identifying multiple people. Although their approach may be suitable for the intended use case, a more general robot application should be able to identify multiple people without the need of any enrollment phase.

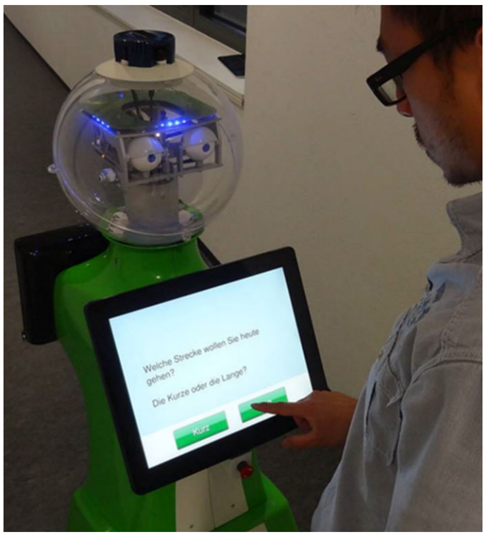

Welche Strecke wollen Sie heute gehen?
Die Kurze oder die Lange?
Kurz

Figure 2.2: The ROREAS patient and walking coach during interaction with a patient. Image from [36].

Service robots often need the ability to follow people [5, 14, 50, 53, 54, 60, 91]. These robots are incorporated with re-identification systems very similar to the ones in the ROREAS and ROGER projects. With the exception of the project by Satake et al. [91], which can identify five individuals, the rest of these systems are only designed to identify one specific person. In all cases these individuals have to be either known in advance, or manually identified in the first video frame, which does not allow the robot to automatically register new people on the fly.

Other researchers incorporate people re-identification in automated push-carts for shopping malls [81], and to give social robots the ability to follow people [15]. These do however only identify one pre-defined person, and are more concerned about addressing short-term occlusion for people tracking [15] and the task of keeping a safe physical distance between the robot and the target [81].

2.3 Computer Vision Person Re-Identification

2.3.1 Definitions

Amongst CV researchers, person re-ID is the task of re-identifying a given person of interest in a collection of images or videos. The person of interest is known as a *probe* person, and its identity is unknown. The collection of images or videos is known as the *gallery*. The gallery is a potentially large database of persons whose identities are known. The underlying assumption in re-ID is that there is guaranteed to exist at least one image or video of the probe person in the gallery. The objective of the re-ID system is to correctly match the probe person with this image or video in the gallery.

More formally, given a query person, or *probe* p, and a database, or *gallery* G, containing $g_{i=0}^{N}$ unique identities, the goal is to find

$$p_x = arg\ max_{n \in 1,2,\ldots,N}\ sim(p, g_n) \tag{2.1}$$

where p_x is the person in the gallery that matches the probe p (same identity), and $sim(p_a, p_b)$ is a similarity function that returns high values for similar input pairs *(person a = person b)* and low values for dissimilar input pairs *(person a ≠ person b)*. In the re-ID literature, the database is commonly referred to as the *gallery*, and the query as the *probe*, and this notation will therefore be subsequently used in this thesis. In CV, re-identification is performed by using visual features derived from the entire human body. The probe and gallery therefore consist of BBs that are cropped tightly around pedestrians. Figure 2.3 is an example of BBs typically found in the probe and gallery. The left hand side depicts a BB in an image, and the right hand side depicts BBs in video (consecutive images).

Figure 2.4 depicts the components in a typical re-ID system. Re-ID systems need two essential components, namely a person *feature vector* (also known as a *appearance model* or a *person descriptor*) and a *distance metric* used to match the feature vectors. The appearance model is a collection of visual, and, in some cases, temporal attributes, or features, that describe a person. Good appearance models are discriminative, meaning that the appearance models of any two different identities should be as dissimilar as possible, while two appearance models of the same person should be as similar as possible. The distance metric is used to measure how similar or dissimilar two appearance models are, and it highly depends on good appearance models to perform well.

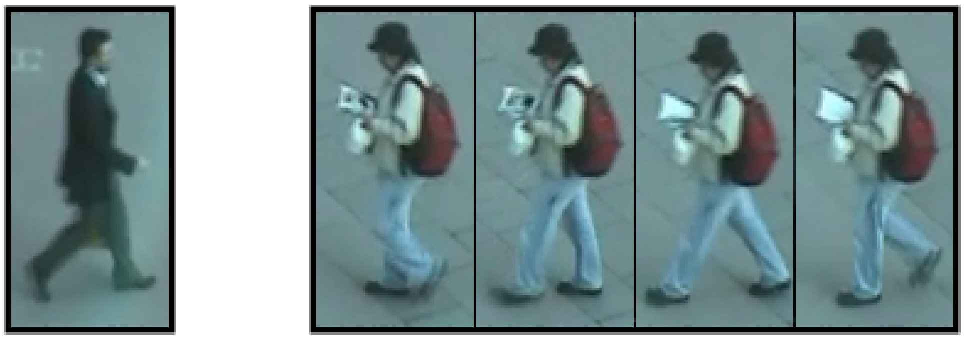


Figure 2.3: Example of bounding boxes around probe/gallery image (left) and probe/gallery image sequence (right). Images are extracted from the PRID-2011 re-ID dataset [41].

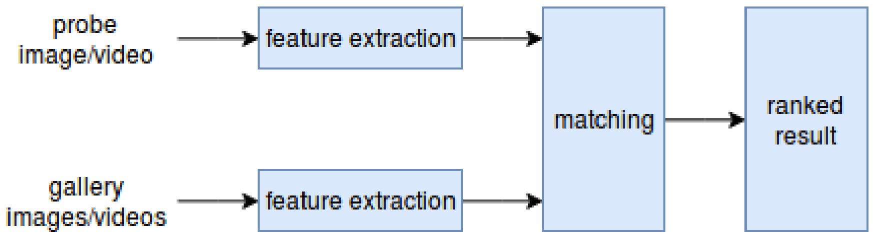


Figure 2.4: The components in a classical re-ID system. The feature extractor extracts features which are used to create an appearance mode of each gallery and probe image/video. The matching component compares the probe appearance model with the appearance models in the gallery, and uses a distance metric to decide which instances in the gallery are most similar to the probe. The matches are sorted by similarity and returned as a ranked list.

For each incoming probe, features are extracted to build an appearance model of this person. Then, the distance metric is used to match the probe appearance model towards each appearance models in the gallery. These matches are sorted by similarity before they are finally returned as a ranked list of identities. In a perfect re-ID system, the true match is found at the top of the ranked list, which describes the match with the highest similarity (known as *rank 1*).

There are numerous factors that differentiate CV re-ID research from the re-ID modules found in the perception systems discussed in the previous section (Section 2.2). While re-ID in a robotic context is a problem of identifying people at different points in time, CV re-ID studies mostly focus on identifying people in networks of two or more surveillance cameras recording data simultaneously or with only short differences in time. Furthermore, robot perception systems integrate person detection, tracking, and re-identification, whereas the majority of CV re-ID research focus solely on the task of person re-identification. Consequently, they commonly assume that the probe and gallery images are perfectly drawn BBs (that is, they are cropped tightly around people containing little background clutter) and that these are readily available beforehand.

2.3.2 Brief History

The term "person re-identification" was introduced in 2005 by Zajdel et al. [129]. The authors developed a mobile robotic vision system that, in addition to performing people tracking, could re-identify people that exited and later re-entered the robot's field-of-view. They used features manually extracted based on the colors of people's clothes and calculated the average of these over time to re-identify people that left and re-entered the field-of-view. Even though they tested their algorithms in a well-lit indoor environment with few people wearing distinctive colors, their algorithms could not adequately handle even slight changes in light conditions (which is bound to happen as robots move around). Their work did however grab the attention of the CV community, which lead to numerous research studies devoted to address the problem of re-ID [134].

Following the approach of the preliminary re-ID work by Zajdel et al. [129], the traditional way to design re-ID systems has been to manually extract features to create person descriptors followed by learning a distance metric to tell them apart. These systems are commonly referred to as *hand-crafted systems*, meaning that features are designed beforehand to retrieve certain data characteristics (as opposed to generic features which will be discussed in Section 2.3.3). After these features are extracted, a distance metric is learned to to tell them apart.

Person Descriptors Based on Hand-Crafted Features

All classification problems rely on good and distinctive features to perform well. In image and video classification, features are derived from the image(s) on the pixel level. In re-ID, features are needed to build person descriptors describing visual properties of people.

Good person descriptors should minimize the *intraclass differences* (i.e. how much variation there is in the descriptors representing the same identity) and simultaneously maximize the *interclass differences* (i.e. how much variation there is between descriptors representing different identities). Having person descriptors with these attributes is important because it directly facilitates the task of learning a distance metric that can distinguish between the different person identities.

Hand-crafted features are derived directly from the pixel values of color images, which are typically represented by the *RGB* color model. In this model, each pixel is represented by three integers, one for the color red (R), one for green (G), and one for blue (B). These integers typically range from 0 to 255. For example, a blue pixel is represented by $[0, 0, 255]$, a black pixel by

[0, 0, 0], and a white pixel by [255, 255, 255]. Although less frequently used in re-ID, other image representations, such as HSV or YUV encoding, can also be used.

Color is the most commonly used feature for pedestrian descriptors [134]. Color is typically used to create a color histogram of the R, G, and B channels in the entire BB, which provides information of the global color distribution. Alternatively, instead of describing the color distribution in the entire BB, histograms can also describe more local areas by for instance segmenting out the person foreground from the background clutter in the BB [27] or specific body parts [21, 67] before creating the histogram. The background clutter in the BB does not provide any discriminative information, and local histograms computed after the background has been segmented out are therefore usually more descriptive. Local histograms do however require more computational resources due to the segmentation and/or body part detection overhead.

Texture features, which aim at describing discriminative edges and patterns in images, can also be derived from the pixel values. The SIFT descriptor [73] is a robust texture-based feature descriptor that is sometimes used in re-ID [133]. However, color features are far more common, presumably because texture features are less suitable to create good appearance models.

Distance Metric Learning

The distance metric is used to determine which person appearance models depict the same identity and which do not. The objective of distance metric learning is to learn a metric that groups data points representing the same class (or person, in this case) close together, while pushing data points representing different classes further apart from each other. In the context of re-ID, the further apart different classes are from each other, the easier it gets to accurately decide the correct identity of incoming probes.

The most common distance metric is perhaps the Euclidean distance which is defined as:

$$d_{Euclidean}(p, q) = \sqrt{\sum_{i=0}^{n}(q_i - p_i)^2} \tag{2.2}$$

where p and q are two feature vectors of length n and $d_{Euclidean}$ is the Euclidean distance between them. From our three-dimensional world, we intuitively think of Euclidean distance between two points as the distance along the straight line that connects them. Although this is the shortest distance in three dimensions, our intuition fails in higher dimensional spaces [25]. Since image features can have a dimensionality of tens or hundreds, more complex distance metrics are necessary. The challenges of dealing with high-dimensional data is known as *the curse of dimensionality*.

Several distance metric learning algorithms have been developed to address higher dimensional data. These often learn a *Mahalanobis* metric, where the goal is to reduce the dimensionality of the feature space such that relevant, discriminate dimensions are kept while less significant dimensions are removed [52]. In the context of re-ID, the KISSME metric learning method [52], which learns a Mahalanobis metric, is the most widely used one [134] because it at least matches the generalization performance of other methods while being significantly faster to train [52]. Although less common in re-ID, support vector machines (SVM) and boosting can also be used to distinguish between different appearance models [34, 132].

Transitioning to Deeply Learned Systems

Deep learning (DL) and convolutional neural networks (CNNs) have gained widespread attention ever since Krizhevsky et al. [57] won the ImageNet[59] image classification challenge by a large margin by using a CNN for feature extraction (see Section 2.3.3 for more details about DL). Following their success, Yi et al. [128] and Li et al. [65] proposed using DL and artificial neural networks (ANNs) for re-ID in 2014. Yi et al. [128] and Li et al. [65] both reported superior performance in terms of accuracy compared to the existing hand-crafted systems. Re-ID systems combining DL and ANNs, which are commonly referred to as *deeply learned re-ID systems*, have been the dominant approach in the re-ID literature ever since [40, 102, 134, 143].

There are mainly two reasons why deeply learned re-ID systems outperform the traditional hand-crafted re-ID systems. First, the deep learning network architectures can learn to extract image and video features that are more robust towards changes in illumination, camera angle and variation in human pose compared to the hand-crafted features. Second, whereas creating person descriptors and learning a distance metric is treated as two separate tasks in the hand-crafted systems, deeply learned systems can jointly extract discriminative features and map the appearance models to a feature space where they are more easily distinguishable. This approach of extracting features and learning a distance metric in an end-to-end manner has shown to be superior compared to treating them as two separate tasks [134]. Deeply learned systems will therefore be the main focus for this thesis.

2.3.3 Deeply Learned Network Architectures for Re-ID

This section presents the fundamental aspects of different DL network architectures (also known as DL *models*) and techniques commonly found in re-ID research. Topics discussed in this section will be put in the context of re-ID starting from Section 2.3.4. Readers may therefore skip to this section if DL for image and video analysis is familiar ground.

Traditional computer algorithms are explicitly programmed with a set of manually constructed instructions, or rules, about how to do a specific task [25]. In some cases, however, manually programming these rules is sub-optimal, and perhaps not not even feasible, and it would be desirable to instead have the algorithm automatically learn them. This is exactly what DL algorithms are responsible for. By looking at data examples, DL algorithms use a training process to observe data patterns that provide useful information for the given task. The more data the DL algorithm is presented with, the more complex models it can potentially learn. The ultimate goal of any DL algorithm is to *generalize* to new, unseen data [25].

There are numerous different types of DL network architectures. Which one to use is problem-dependent, and choosing the correct one is essential to achieve good results. In the domain of image and video analysis, CNNs and *recurrent neural networks (RNNs)*, which are both types of ANNs, are most broadly used. These networks require a learning method. In re-ID, *supervised learning* and *semi-supervised learning* are the most widely used learning methods, but some recent work also explore using *unsupervised learning methods*.

The Learning Process

The learning process is an iterative procedure that adjusts the DL model parameters based on the difference between the predicted classification values and the correct classification values. The combination of all the model parameters is called the *hypothesis function*, and the objective of the learning procedure is to guide the hypothesis function toward a local (or preferably global) optimal solution (also known as a global/local *minimum*).

Learning can be performed either in a supervised, unsupervised or semi-supervised manner [92]. Supervised learning is the task of learning a hypothesis function that maps input to output by training on known input-output pairs [90], and therefore requires labeled datasets. Unsupervised learning uses uncategorized data to learn common data patterns, and is often used when there is no labeled data available. Semi-supervised learning can be placed in-between supervised and unsupervised learning, because it takes use of both labeled and unlabeled data to learn.

The process of learning consists of two steps, the *forward pass* and the *backward pass*. During the forward pass, the data is fed into the network input layer. The values in the input layer are then multiplied with the weights connecting it with the first hidden layer. Then, the data passes through an *activation function* before it gets multiplied with the next weights. This procedure is repeated until the output layer has been reached. The values in the output layer should ideally be as close to the ground truth as possible.

In the backward pass, the network parameters are updated according to an *optimization algorithm* whose goal is to minimize the error between the actual output value and the desired output value. The optimizer often uses a method known as *backpropagation* to compute the gradient of the network. By updating the weights in a manner that follows the gradient in a negative direction, the hypothesis function is guided toward a local or global minimum.

Artificial Neural Networks

ANNs are computing systems whose design draw inspiration from the biological neural networks found in animal and human brains. ANNs consist of *nodes* that are organized in *layers*. The layers are interconnected by *weights* that allow data to flow through the network. These weights are *trainable parameters* whose values form the hypothesis function. The weights are typically randomly initialized following a normal distribution, and multiplied by a factor depending on the size of the layer. As the network is trained following one of the aforementioned learning methods, the weights are adjusted, leading the hypothesis function towards producing more optimal outputs.

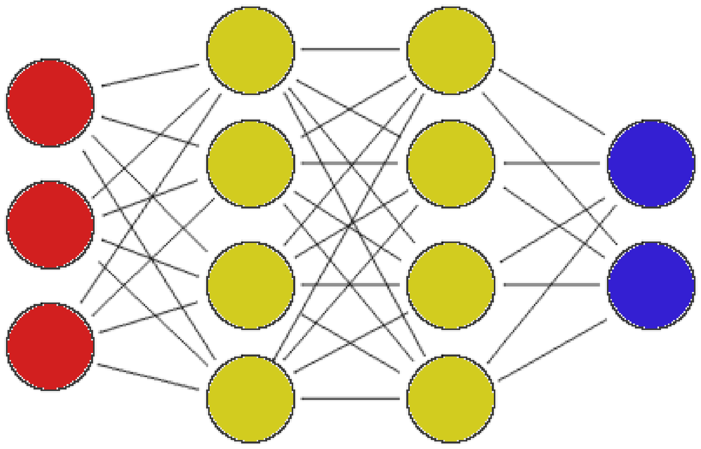


Figure 2.5: A shallow ANN (*depth* = 3) with a fully connected network structure. The network contains an input layer with three nodes (red), two hidden layers with four nodes each (yellow), and finally an output layer with two nodes (blue).

Image 2.5 depicts a *fully connected* (FC) ANN with two hidden layers. (Fully connected means that all nodes in any layer are densely connected to all nodes in the previous and next layer.) In this case, the data is represented by three values (hence three input nodes), and the network outputs two values (hence two output nodes), which typically reflect the number of total classes in a classification problem. The number of hidden layers, known as the network *depth*, along with the *width* of each hidden layer, are *hyperparameters* that must be specified in advance. In general, more complex problems require deeper and wider network architectures. Deeper and wider networks do however have more trainable parameters, which increases the training time and run time in addition to the risk of *overfitting* to the training data. Overfitting means that the network learns the training data perfectly, but is unable to generalize to new, unseen data. It is therefore often desirable to keep the network structure as shallow as possible without neglecting accuracy.

Convolutional Neural Networks

Although FC ANNs are powerful for certain tasks, they are not capable of picking up the spatial information in images and videos. Images typically contain hundreds of pixels in the vertical (H) and horizontal (W) directions, each of them consisting of three RGB values. This consequently results in a high amount of input values to the neural network ($3 \cdot H \cdot W$). Traditional FC networks are not well-suited for this type of data because 1) the high amount of input nodes results in a lot of trainable parameters and 2) the one-dimensional layer structure is unable to pick up the three-dimensional spatial information that images contain. As a result, a more appropriate network structure for analyzing images are CNNs.

CNNs differ from traditional FC ANNs in that each layer in the network is built up of filters, or *kernels*, instead of fixed weights, and the nodes are replaced by three-dimensional *feature maps*. These layers are known as convolutional layers, and contain most of the learnable network parameters. The kernels slide across the feature maps of the previous layer (or across the input image in the first layer), performing a *convolution* at each location, which creates the feature maps for the next layer. One single convolutional layer takes a three-dimensional volume as the input and produces a three-dimensional volume as the output [47]. The number of kernels and their sizes control the depth, and to some degree the height and width, of the output volume. It is common to have fewer, but larger, kernels in the first layers, and more, but smaller, kernels in the final layers [38, 57, 96, 101]. As can be seen in Figure 2.6, the depth of the feature maps often increases throughout the network while the width and height decreases.

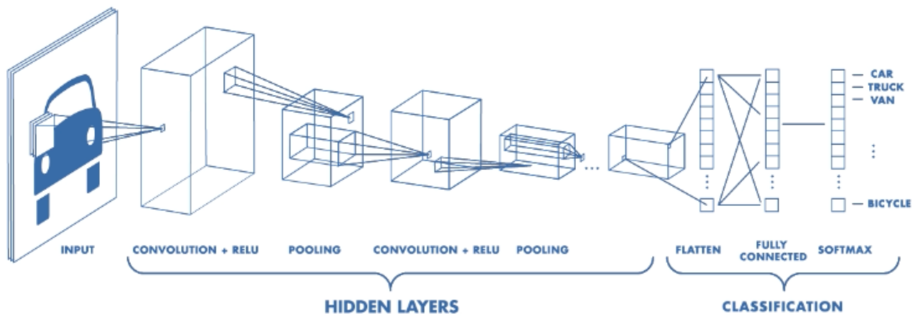


Figure 2.6: An example of a CNN architecture with convolutional layers for feature extraction, ReLU activation function for non-linear mapping, pooling layers to reduce spatial dimensionality and the amount of parameters, and finally a flattened fully connected classification network using the softmax activation function to perform the classification [84].

In CNNs, the convolutional layers contain most of the learnable network parameters, and are therefore arguably the most essential part of the network. The convolution is however simply a linear operation. A network only consisting of linear operations would only be able to learn a linear mapping from input to output, which would highly restrict network's learning ability, especially in complex problems.

To introduce non-linearity to the model, an *activation function* is applied on the feature maps immediately after the convolutional layer. This converts the feature maps to *activation maps*, though these two terms often are used interchangeably. Many different activation functions exist, and which to use highly depends on the network architecture and the problem the network is used for.

In addition to the convolutional layers and activation functions, CNNs contain *pooling layers* that are commonly inserted periodically between convolutional layers. Pooling layers reduce the spatial size of the network, which reduces the amount of parameters and number of computations [47]. Equivalently with the convolutional layers, pooling layers consist of filters that perform local operations on the input volume. Pooling layers do however not contain any trainable parameters. Furthermore, pooling-layers operate on each depth of the activation maps independently instead of operating across the entire depth at once.

Although a variety of different pooling layers exist, the *max-pooling* layer is most frequently used [47]. As the kernel slides over the activation maps, the max-pooling operation keeps the highest value and discards all other values. Parameters such as kernel height, kernel width and *stride* (how many pixels the filter is moved after each pooling operation) need to be specified in advance. An illustration of the max-pooling operation can be seen in Figure 2.7.

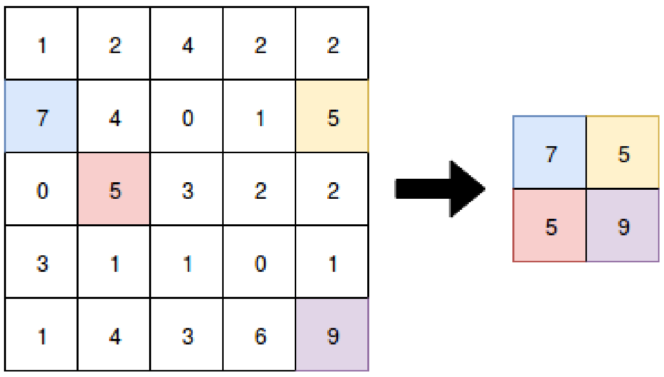


Figure 2.7: Max-pool operation with stride = 2 and a 3×3 filter sliding over a 5×5 activation map (left), resulting in a 2×2 output volume (right). Best seen in color.

In a classification problem, a shallow FC ANN is often inserted after the final convolutional layers. There are mainly two reasons for using FC layers in CNNs. First, the fact that FC layers have connections to all neurons in the previous layer enables the network to mix features from all image regions when making decisions. Second, FC layers can efficiently reduce the feature dimensionality to a one-dimensional output vector, rendering well-suited for training classifiers. The right-hand side of Figure 2.5 depicts two FC layers that are connected after all convolutional layers.

Recurrent Neural Networks

While CNNs are excellent to extract spatial features in single images, they are not designed to discover temporal features found in sequential data such as text strings, speech or video. RNNs, however, are another type of neural networks more suited to extract temporal features. RNNs can discover these temporal cues because the network architecture unfurls over time, keeping connections to the node found at the previous time step. In contrast to CNNs, RNNs can thus combine information from the current and previous time steps to make decisions. RNNs are also trained in a supervised manner using gradient descent to minimize the network error. To update the weight parameters, backpropagation through time (BPTT), which is a modification of the backpropagation algorithm used in FC ANNs and CNNs, is commonly used.

Deep Learning for Classification

Classification is a problem that often can be approached using ANNs. The goal is to learn a hypothesis that can assign the correct class (or label) to the input data. Classes are commonly represented by a *one-hot* encoded vector that contains the number one at the position representing the correct class, and zero at all other positions. Given any input data, the hypothesis should output a value close to one at the correct index, and values close to zero at all other indexes. The classification problem goes hand in hand with supervised learning when a one-hot labeled training dataset is available. As an example, a dataset with four classes: car, truck, van, and bicycle, can be represented as a four-dimensional vector $['car', 'truck', 'van', 'bicycle']$. A data point belonging to the class truck would then have the ground truth values $[0, 1, 0, 0]$, and a good hypothesis should output values as close to this as possible.

One important observation to make is that the number of different classes has to be known when designing the network structure. For instance, a network with two output nodes, such as the network depicted in Figure 2.5, is only suitable for a classification problem where there are two classes. Furthermore, the number of different classes has to be static and decided in advance, because changing the amount of classes would require a different network structure.

Siamese Network Structure

In some cases, due to the nature of the problem or the lack of labeled data, using the one-hot classification approach may not be feasible. An alternative approach is to train the network by presenting pairs of similar or dissimilar data points. This allows the network to learn discriminative features that can be used to decide whether two new data points represent the same class or not.

A common approach to train on pairwise data is to use the *Siamese network architecture*. A Siamese network consists of two or more separate but identical neural networks that share the same weight parameters. These networks could in theory be of any kind, but in re-ID they are usually CNNs [2, 79, 85, 95, 110, 112, 128, 143] or sometimes CNNs and RNNs combined [8, 78, 107]. The key point, however, is that the networks do not have the final FC classification layers. Instead, the features generated by the CNN or RNN can be pairwise compared in order to separate the similar identities from the dissimilar ones. The Siamese network learns an embedding that pulls similar data close together and pushes dissimilar people further apart in the feature space.

Instead of directly using class information to train, the network only needs to know if the input data depicts the same person or not. Consequently, the Siamese model can handle an arbitrary amount of classes. This is a huge advantage when the number of classes is either very large, of unknown size or when it varies over time.

The backpropagation algorithm is applicable to train a Siamese network, but instead of comparing the network output against a ground truth vector, the loss function is based upon how similar or dissimilar the two output feature vectors are. This training method does not take full advantage of the available labels, and is therefore known as semi-supervised learning.

2.3.4 Image-Based Re-ID

Re-ID research is often divided into two different categories, namely *image-based re-ID* and *video-based re-ID* (see Section 2.3.5). In an image-based re-ID approach, the probe p is an image of a person and the gallery G is a collection containing one image of every person in the model. The majority of current re-ID research and available dataset fall into the image-based category. The image-based problem can either be treated as a classification problem or as a matching problem.

The Classification Approach

In the classification model, each person in the gallery is treated as a separate class, and the objective is to assign the correct class to each and every probe image. The ground truth is one-hot encoded, thus it highly resembles the problem of image classification. However, the fact that datasets contain many identities (ranging from tens to over a thousand), contain few images of each identity (often as little as two) captured from different perspectives, makes it a challenging classification task.

Variations of CNNs with FC classification layers are commonly used in the classification approach [61, 64, 99, 124, 141]. Su et al. [99] and Li et al. [61] detect body parts in separate sub-networks that are fused together to generate the combined classification output. Li et al. [64] have a somewhat similar approach, and derive both local features from selected image patches in addition to global features from the entire BB. Xiao et al. [124] combine data from six datasets to form a large gallery of identities allowing their network to learn features from the different domains. Zheng et al. [141] combine a classification CNN with a Siamese model, resulting in a network that can simultaneously classify each identity and compute similarity scores between image pairs.

The major drawback with the classification approach is that the number of different classes depends on how many identities there are in the gallery. These networks are therefore only suitable for the specific dataset they were trained on, and are unable to tackle the dynamic gallery size in real-world applications. This drawback can be overcome by replacing the final classification layer with a distance metric during inference time. The distance metric would then essentially use the features learned by the classification network during training to decide if two features represent the same person or not. This approach is however sub-optimal because the network and the distance function are two completely separate blocks, meaning that it is challenging to find the distance metric best suitable for these specific feature representations.

The Pairwise and Triplet-Wise Matching Approach

Rather than assigning one class to every person, the matching approach uses a Siamese network to compare pairs or triplets of images to decide if these images depict the same person or not. The only information needed to train such an architecture is whether the two images depict the same person or not, which leaves it as an attractive option for the re-ID problem.

The major advantage of this approach is that it does not rely on a fixed number of classes. The Siamese architecture can thus handle a dynamic person gallery of arbitrary size both during training and run-time. In most cases, this network attribute makes it a much more viable option for real-world applications compared to the classification approach. The downside is that the number of comparisons needed during run-time grows linearly with the gallery size. This disadvantage is however arguably negligible unless the gallery is very large.

Many researchers have studied re-ID using the pairwise [2, 79, 85, 95, 107, 110, 112, 128, 143] or triplet-wise [8, 16, 40, 69, 70, 102, 110] matching approach. These works use a Siamese architecture consisting of CNNs or a combination of CNNs and RNNs to distinguish similar from dissimilar identities.

Although these works report promising performance on datasets, none of them are tested on real-world applications. These systems commonly assume that the query person is guaranteed to be found in the gallery, whereas a re-ID system in the context of robotics will have to deal with a dynamic amount of both known and unknown individuals. This assumptions leave them unable to meet the requirements of a re-ID system for a mobile robotic platform.

Motivation for Video-Based Re-ID

Practical applications both in surveillance and robotics mostly use video cameras instead of still images. Using only images for re-ID is suboptimal because it does not take advantage of the rich temporal information found in videos. The upcoming section looks into the advantage of exploiting temporal cues in the re-ID problem.

2.3.5 Video-Based Re-ID

Video-based re-ID systems utilize videos (or multiple consecutive image frames) rather than single images to identify people. Equation 2.1 still applies, but now the probe p is a video sequence and the gallery G contains one or several image sequences, rather than single images, of each person. This distinction is illustrated in Figure 2.3.

Video data provides richer information than single images. Characteristic temporal cues, such as gait patterns, can supplement spatial features to disambiguate difficult cases. Combining spatial and temporal (spatio-temporal) features, is known to perform better than system solely relying on spatial features [134]. Additionally, video re-ID is more relevant for real-world applications as people will typically be observed in video cameras [78].

One approach is to use the Siamese model from image-based re-ID and pooling the results to obtain one feature vector per video sequence [40, 61]. This approach is fairly straightforward with little overhead compared to the purely image-based counterparts, but still shows promising performance on various benchmarks.

Other researchers incorporate RNNs in the Siamese pipeline [71, 78, 82, 119, 125]. As expected, the RNN's superior ability to pick up temporal features result in better performance. Although not reported, this is likely to reduce efficiency somewhat due to the added system overhead.

In an effort of avoiding the need of large-scale datasets, some researchers attempt at training in a more unsupervised manner by using only one labeled video per identity [72, 120, 127]. While this approach has the potential to save a lot of manual labeling work, the re-ID accuracy is currently significantly worse than the supervised counterparts.

Despite the fact that the majority of video-based re-ID systems report better accuracy than the image-based approaches, they all assume that the gallery size is fixed and known in advance, which leaves them unsuitable for mobile robotic applications.

2.3.6 Toward More Practical Re-ID Systems

Some works attempt to minimize restrictive assumptions that separate re-ID research from real-world applications. These can be divided into two categories: *open-world re-ID* and *end-to-end re-ID*. Open-world re-ID approaches the re-ID problem in a setting where the probe identities are no longer guaranteed to exist in the gallery. End-to-end re-ID aim at combining automatic person detection and re-identification to one system instead of relying on the labeled BBs given in the datasets.

Open-World Re-ID

In an open-world re-ID setting, the identities of the probe persons are no longer guaranteed to be present in the gallery. As can be seen in Figure 2.8, the open-world setting requires an extra component to determine if the probe and the top gallery matches are more similar than a given threshold. This setting was initially studied by Liao et al. [68], where the performance of several metric learning algorithms were evaluated. The authors' best reported identification rate was as low as 17% on their newly proposed evaluation metrics for the open-world re-ID setting, indicating that the open-world scenario is significantly more challenging than traditional re-ID. While Zheng et al. [140] reported slightly better performance on public datasets, their experiments confirmed the challenging nature of open-world re-ID.

More recently, Zhu et al. [144] investigated the open-world problem in a large-scale setting. They exploit positive identity pairs to extract discriminative features, and use hashing to group similar people together in the search space. Their experiments show significant improvements in terms of accuracy compared to the previous works [68, 140]. Although efficiency is not reported in terms of seconds or FPS, the hashing function reportedly allow for a search time of at least a order of two magnitudes faster than non-hashing re-ID methods.

Efficient search is crucial in any surveillance system, especially when mounted in public places, as the cameras continuously produce large amounts of data. This is however slightly less relevant for a mobile robotic application as the robot is likely to encounter fewer, and often the same, people.

All re-ID research in the open-world setting to date currently falls in the image-based category [68, 140, 144], and is therefore not adequate for mobile robotic applications.

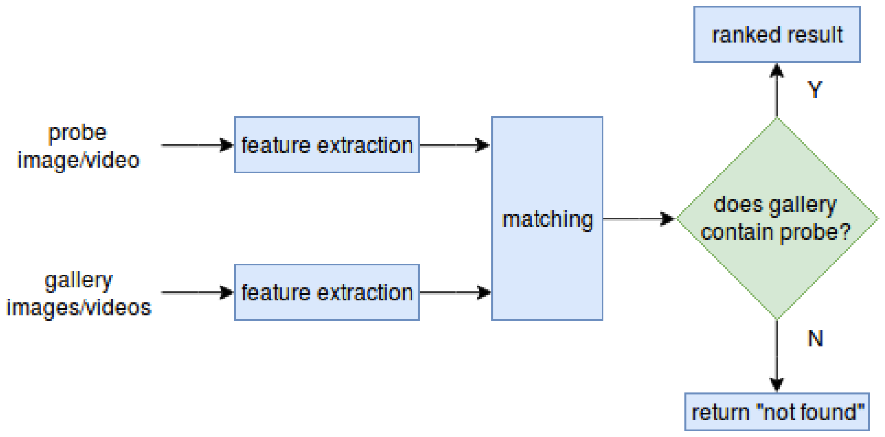


Figure 2.8: The components in an open-world re-ID system. In addition building and matching appearance models (traditional re-ID), the open-world re-ID systems additionally check whether or not the probe identity exists in the gallery.

End-To-End Re-ID

By combining person detection and re-ID, end-to-end systems aim at making re-ID more applicable for real-world applications. As depicted in Figure 2.9, these methods typically rely on automatic people detection algorithms, which makes the re-ID problem harder for two reasons. First, the people detection accuracy directly impacts the re-ID accuracy [123, 126, 136]. More accurate detectors will detect less outliers, i.e. various objects wrongly classified as people, but this typically requires more computational resources. Second, since the gallery is formed by the detected pedestrians, the detection accuracy directly impacts the gallery size. Due to the dynamic gallery size, Equation 2.1 (where it was assumed that p is guaranteed to be found in G) no longer applies.

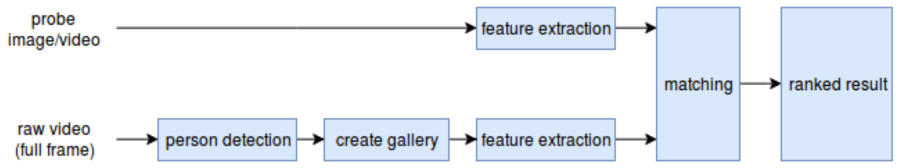


Figure 2.9: The components in an end-to-end re-ID system. In addition to the feature extraction and matching blocks found in the classic re-ID approach, an automatic person detector is used to form the gallery.

There are only a handful of projects that focus on end-to-end re-ID. Xiao et al. [123] and Zheng et al. [136] both developed end-to-end re-ID systems along with releasing large-scale datasets (see Chapter 3 for an overview of public dataset). Both projects rely on automatic pedestrian detection to generate the gallery, and Zheng et al. [136] additionally rely on detectors to generate probe images. Both methods report that the the detector quality has a high impact on re-ID accuracy.

Even though end-to-end re-ID pushes traditional (and also open-world) re-ID closer to real-world scenarios, the projects by Xiao et al. [123] and Zheng et al. [136] fall under the image-based re-ID category, and are thus not adequate for real-world robotic applications.

2.3.7 Multiple Object Tracking

Multiple Object Tracking (MOT)[1] is a separate CV research domain concerned about tracking multiple objects and people simultaneously. Every tracked person is assigned an identity, and one objective of the tracker is to re-assign the correct identity after a person has been temporarily out of sight. Temporary occlusions, which are frequently caused by foreground objects and crowded scenes, make the tracking problem very challenging. To handle the challenging scenarios caused by occlusion, many recent trackers use deeply learned networks trained on re-ID datasets to assign a person appearance model to each tracked person [109, 111, 118, 122]. The appearance model is used to reassign the correct identity to people after they have been occluded.

Nevertheless, these projects are only concerned with short periods of occlusions typically lasting for less than a second. Very short-term occlusions induces little variation in the background and light conditions, changes in human pose and varying camera angles, and identification systems designed for these situations are likely to perform poorly on mobile robotic applications.

2.3.8 Long-Term Re-ID

Most state-of-the-art re-ID research only considers very short-term re-ID. On a real-world application, however, people are likely to be absent for minutes, hours, or even days before re-entering a surveilled area or a mobile robot's environment. This implies further complications such as drastic illumination changes, pose variations, and different colored and styles of clothing.

Zhang et al. [131] recently took the initial step towards studying re-ID in a long-term setting. Arguing that color features alone are not sufficient due to the added challenges, their approach belongs to the video re-ID category, exploiting temporal information extracted from people's trajectories.

While their long-term re-ID study is more applicable for most real scenarios, Zhang et al. [131] rely on the same restrictive assumptions as many of the other re-ID projects. They use hand-drawn BBs, static

[1]MOT has its own yearly tracking challenge: https://motchallenge.net/

surveillance cameras, and assume that the gallery contains the query person. None of these assumptions are reasonable for most mobile robot applications, leaving longer-term re-ID for real-world applications an open problem.

2.3.9 System Efficiency

System efficiency in terms of FPS is of high priority for any robotic system that need to operate in real-time. In the CV community, which includes research in re-ID, efficiency is however often neglected to achieve higher accuracy. In re-ID, few papers mention efficiency, meaning that most re-ID systems are likely to be of little practical value for robotics.

A popular trend in re-ID is to apply metric learning on deep, pretrained networks, such as AlexNet [57], VGG-16 [96], GoogLeNet [101], or ResNet [38]. These networks have typically been trained on large-scale image datasets and later fine-tuned on re-ID dataset. While this approach has proven to be promising on re-ID benchmarks, networks with deeper architectures require more computational resources, and are therefore likely to exceed the real-time requirement found in robotics.

2.4 Summary

There have been many efforts in addressing the challenging problem of people re-identification in the fields of HRI, robotics, and CV. In CV re-ID systems, the assumptions of fixed (and known) gallery size and perfectly hand-drawn pedestrian BBs are the main factors that hinder adaptation to real-world robotic applications. Furthermore, the majority of these re-ID systems create appearance models based on images, leaving them unsuitable to re-identify people in video data. The lack of documented system efficiency also gives reason to believe that many re-ID systems do not reach the real-time requirements in robotics.

In HRI and robotics, most systems require having person appearance models available in advance and are often limited to identifying only one individual at a time. While some robotic re-identification systems can re-identify multiple people without prior appearance knowledge [113, 115], they use biometric features which requires people to directly face the robot at an appropriate distance. This is not convenient because it requires robots to interfere with the activities of the humans and may result in very artificial and unnatural interaction between humans and robots.

Furthermore, with the exception of systems relying on biometrics (such as face cues), re-identification is mostly done over a very short time span. To adapt to dynamic human-centered environments and build personal relationships, robots need to be able to re-identify people over longer time periods [58]. Re-identification over several minutes, hours, or even days is more challenging because it implies large variances both in light conditions and person appearances.

To overcome these shortcomings in CV re-ID, the proposed models are trained using a dataset whose BBs are not perfectly hand-drawn, but rather detected in an automated manner (see Section 3.2 for details about the selected dataset), meaning that the re-ID system developed in this thesis is tailored for a robotic context. Experiments are conducted with varying gallery sizes to simulate various real-world robotic scenarios (Section 5.3.3), and various ways of using video data instead of images are examined (Section 5.2.2), which renders the re-ID model compatible to process video data captured by mobile robots. Finally, the processing speed is evaluated in terms of frame rate (Section 5.3.4), where it is shown that it is possible to obtain real-time performance without neglecting accuracy.

Furthermore, the proposed re-ID system does not require having person appearance models available in advance and is not limited to only re-identifying a small amount of people. The model does not rely on biometric features and can re-identify people under large changes in camera angles and lighting conditions, meaning that this work develops a system that is capable of re-identifying people in a manner that promotes natural and pleasant interactions between humans and robots.

Chapter 3

Datasets

This chapter presents an overview and comparison of publicly available re-ID datasets. Important factors to take into consideration when selecting appropriate datasets are examined. Finally, various aspects of the dataset selected to evaluate the proposed re-ID system are discussed.

3.1 Dataset Considerations

There is a large amount of publicly available labeled re-ID datasets. Table 3.1 shows an overview of the most popular datasets along with some of their attributes [31].

Labeled datasets are required to train any deep ANN in a supervised manner. There are multiple factors that need to be taken into consideration when selecting an appropriate dataset. For a robotic application, a dataset that reflects the typical circumstances of a mobile robot is required. This thesis therefore identifies and assesses three dataset attributes that are of high importance for the re-ID model: *dataset size* (Section 3.1.1), *data type* (Section 3.1.2), and *data labeling method* (Section 3.1.3). Furthermore, an appropriate dataset needs to contain video sequences of entire persons (not only their faces) and have large variation in people appearances. Additionally, the data needs to be captured from multiple angles with ego-centric points of view to mimic the data captured from sensor mounted on mobile robots.

3.1.1 Dataset Size

Complex classification problems with many classes require larger datasets to train a deeply learned model. A large amount of data is needed to learn more discriminative features that can better represent each person. If the dataset is too small, on the other hand, the model will not have enough data to train, leaving it unable to learn discriminative features and to generalize to unseen data.

In 2014, Li et al. [66] published the CUHK03 dataset, which was the first re-ID dataset that was large enough to train a deeply learned classification model. The CUHK03 dataset contains over 13 000 BBs of 1 467 different people. DL for image analysis became the dominating approach in re-ID after Li et al. [65] and Yi et al. [128] successfully trained deep CNNs using the CUHK03 dataset. Consequently, as can also be seen in Table 3.1, re-ID datasets published after 2014 are significantly larger both in terms of the number of people and the number of image frames compared to older datasets. The increase in re-ID dataset size and the ever-progressing performance of ANNs for image classification and feature extraction has lead to the integration of ANNs in re-ID algorithms being the primary approach in recent research.

3.1.2 Data Type

In re-ID, there is a clear distinction between image-based datasets and video-based datasets. Image-based datasets typically consist of two or more *images* of each person captured at different times and/or from different camera angles (see Figure 3.2). Video-based datasets, on the other hand, consist of two or more *videos* (often of varying length) of each person captured at different times and/or from different camera angles (see Figure 3.3). Using

Dataset	Year	#people	#BB	Detection	Tracking	Data type	FP?	Environment	Evaluation
VIPeR [33]	2007	632	1 264	hand	n/a	RGB image	no	-	CMC
ETHZ [94]	2007	148	8 580	hand	hand	RGB video	no	city/ street	CMC
iLIDS [139]	2009	119	476	hand	n/a	RGB image	no	airport	CMC
GRID [74]	2009	250	1 275	hand	n/a	RGB image	yes	subway	CMC
CAVIAR [17]	2011	72	610	hand	n/a	RGB image	yes	mall	CMC
PRID2011 [41]	2011	200	24 543	hand	hand	RGB video	yes	campus	CMC
3DPES [9]	2011	200	1 011	hand	hand	RGB video	no	campus	CMC
PAVIS [10]	2012	79	-	Microsoft Kinect SDK	n/a	RGB-D & skeleton image	no	indoor	-
WARD [77]	2012	70	4 786	hand	n/a	RGB image	no	-	CMC
CUHK01 [63]	2012	971	3 884	hand	n/a	RGB image	no	campus	CMC
CUHK02 [62]	2013	1 816	7 264	hand	n/a	RGB image	no	campus	CMC
CUHK03 [66]	2014	1 467	13 164	hand/ DPM [28]	n/a	RGB image	no	campus	CMC
RAiD [22]	2014	43	1 264	hand	n/a	RGB image	no	campus	CMC
BIWI [80]	2014	50	-	Microsoft Kinect SDK	-	RGB-D & skeleton video	no	indoors	-
iLIDS-VID [114]	2014	300	42 495	hand	hand	RGB video	no	airport	CMC
Market-1501 [137]	2015	1 501	322 010	hand/ DPM [28]	n/a	RGB image	yes	campus	mAP&CMC
MARS [135]	2016	1 261	1 191 003	DPM [28]	GMMCP [23]	RGB video	yes	campus	mAP&CMC
LSPS [123]	2016	11 934	34 574	hand	n/a	RGB image	-	city & movies	mAP&CMC
PRW [136]	2016	932	34 304	hand	n/a	RGB image	yes	campus	mAP&CMC
DukeMTMC-reID [142]	2017	1 812	34 441	hand	n/a	RGB image	yes	campus	CMC
DukeMTMC-4ReID [32]	2017	1 852	46 261	Doppia	n/a	RGB image	yes	campus	CMC
Florence 3D [83]	2018	16	39 315	-	-	RGB-D & skeleton video	no	-	CMC
RPIfield [138]	2018	112	601 581	ACF [24]	IoU	RGB video	yes	campus	-
MSMT17 [116]	2018	4 101	126 441	Faster RCNN [87]	n/a	RGB image	no	campus	mAP&CMC

Table 3.1: Some statistics of popular re-ID datasets that have been taken into consideration for this project. A hyphen (-) means unspecified or unknown.

video sequences opens the possibility to exploit both temporal and spatial features, such as gait and various motion patterns.

Some datasets consist of color images/video (RGB) combined with depth images/video (RGB-D). RGB-D data is captured by cameras with an additional sensor that produces depth images where each pixel represents the physical distance between the camera and the object it depicts. Depth data is typically used to supplement RGB data in image and video analysis.

The Microsoft Kinect [1] is a popular RGB-D camera that is frequently used in robotics. Skeleton data, which is information about position and length of various limbs and body parts, is often provided along with the RGB-D data because the Microsoft Kinect Software Development Kit (SDK) allows for easily extracted skeleton data. Figure 3.1 depicts the RGB picture, depth picture and the corresponding skeleton data from the BIWI re-ID dataset [80].

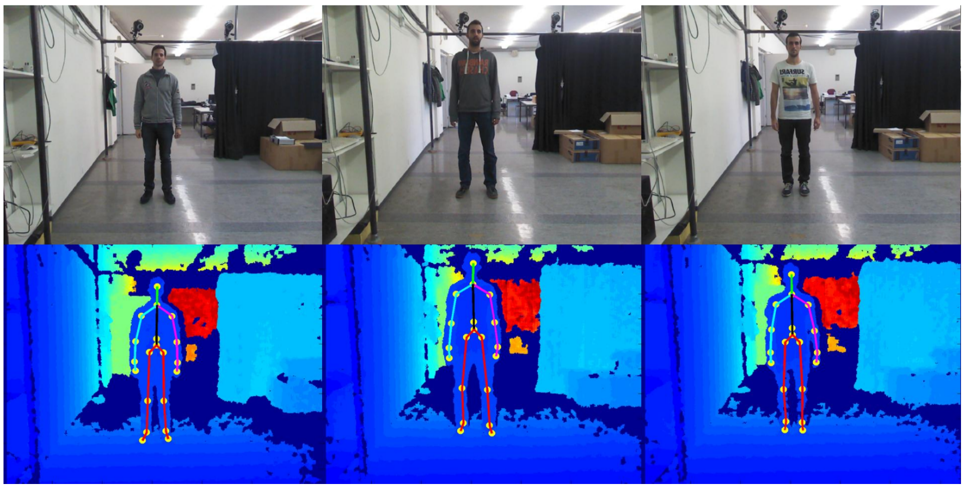


Figure 3.1: Images from the BIWI re-ID dataset [80] with RGB images (top row) and corresponding depth images with visualization of the extracted skeleton information (bottom row).

Since mobile robots typically have RGB or RGB-D video sensors, a model trained on a video-based dataset is likely to generalize better than a model trained on still images. While depth data can provide useful information that may simplify or improve the re-identification accuracy, current RGB-D re-ID datasets, the PAVIS dataset [10] depicting 79 people, the BIWI dataset depicting 50 people [80] and the Florence 3D dataset [83] depicting 16 people, are not large enough to train a deeply learned model. Furthermore, depth sensors often only work indoors, and the depth data can only be measured accurately in a certain distance range (50cm to 5m for the Microsoft Kinect) [1]. Processing depth data also requires a fair amount of computational resources, which are usually limited on mobile robots, and processing depth images should therefore only be used if it significantly improves efficiency or accuracy.

[1]https://developer.microsoft.com/en-us/windows/kinect

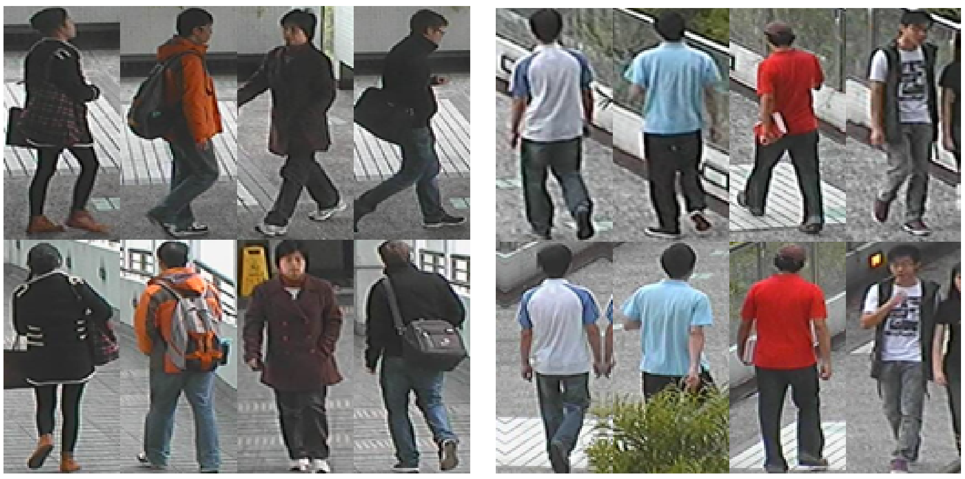


Figure 3.2: Typical image re-ID dataset containing pairs of bounding boxes of each person. Images are taken from CUHK01 [63] (four left) and CUHK02 [62] (four right). Vertically aligned images represent the same person.

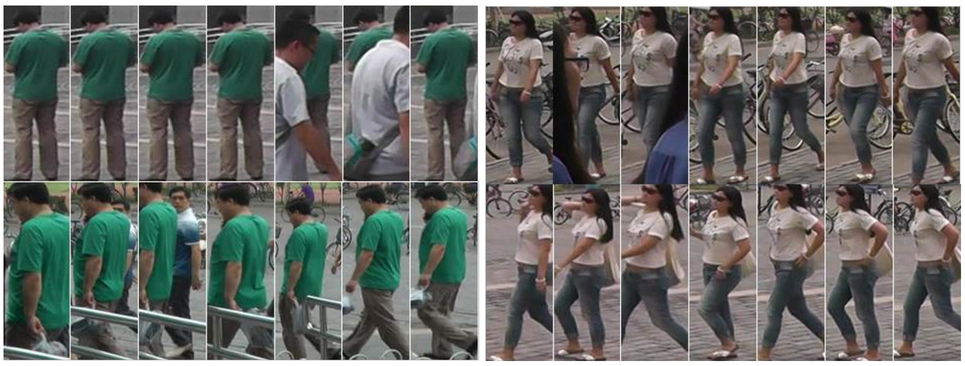


Figure 3.3: Typical video re-ID dataset containing multiple consecutive frames for each person. Images are taken from the MARS dataset [135]. Vertically aligned sequences represent the same person.

3.1.3 Labeling Method

In re-ID datasets, each image is commonly labeled with a detected BB around the depicted person along with the person's unique identification number. In video-based datasets, tracking is an additional labeling step that associates the BBs of each person in consecutive frames. In the context of re-ID, the labeled BBs are directly used to train the model, which means that the method used to annotate the dataset will affect how well the model generalizes to new data.

Labeling is most commonly performed by manually drawing the outline of the BBs around each person. Manual hand-labeling techniques are very time consuming, but typically result in accurate annotations. An alternative to manual labeling is to use off-the-shelf algorithms to produce the dataset annotations. This approach is more efficient, but often results in poorer labeling quality in terms of accuracy. For instance, automatic detectors and trackers will produce false positives (FPs), also known as *distractors*, which is background clutter or other objects wrongly detected/labeled as people. Furthermore, BBs produced by automatic detectors and trackers are not perfect, and will often either be slightly too small, too large, or skewed towards one side.

Any real-world re-ID system will need to rely on automatic person detectors and trackers to produce the BBs, meaning that FPs and inaccuracies are guaranteed to occur. Despite the presence of BB inaccuracies in datasets that have been labeled following an automated fashion, these datasets have a closer resemblance to the data in any real-world re-ID system. It is therefore argued that training a model on a dataset whose BBs have been automatically labeled is likely to perform better than a model trained on a hand-labeled dataset when deployed on a mobile robot. Datasets with hand-drawn BBs have therefore been excluded for this project. Note that even though the BBs have been produced by an automatic person detector, the ground truth person identity still needs to be manually annotated.

3.1.4 Conclusions

After evaluating the different datasets by taking dataset size, data type, and labeling method into account, the Motion Analysis and Re-identification Set (MARS) [135] was selected as the most appropriate dataset to train and evaluate the re-ID model. While it would have been desirable to have a dataset with depth data, there are currently no depth datasets sufficiently large to train a deeply learned re-ID model. The following section (Section 3.2) looks into the details of the MARS dataset.

3.2 The MARS Dataset

3.2.1 Dataset Attributes

The MARS dataset [135] is the largest existing video re-ID dataset. MARS has a total of 1 191 003 video frames depicting 1 261 different people. There are on average 13 video sequences, or *tracklets*, per person, each of them averaging 59 video frames. These sequences are captured by six cameras at a university campus in Beijing. Figure 3.3 depicts four video sequences of two different people in the dataset.

Instead of manually annotating the BBs around each person, which is the most commonly used annotation method in re-ID datasets, the creators of MARS have done the annotation in an automated manner by using the Deformable Part Model (DPM) [28] person detector and the GMMCP [23] person tracker. Automatic labeling generally introduces more error than hand-labeled datasets, making the re-ID problem more challenging. On the other hand, automated annotation is likely to be similar to what can be expected to be available on a mobile robotic application, as robots also rely on automatic person detection and tracking. Furthermore, as depicted in

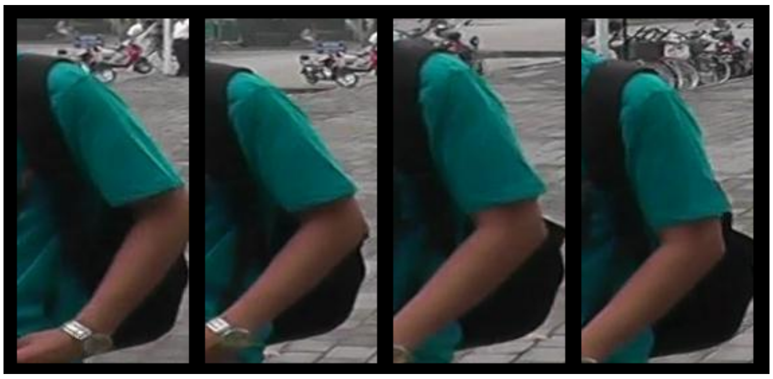


Figure 3.4: An example of a distractor sequence from the MARS dataset [135]. While some distractors are video sequences that only depict a small part of a person, other distractors may be videos of the background or other objects.

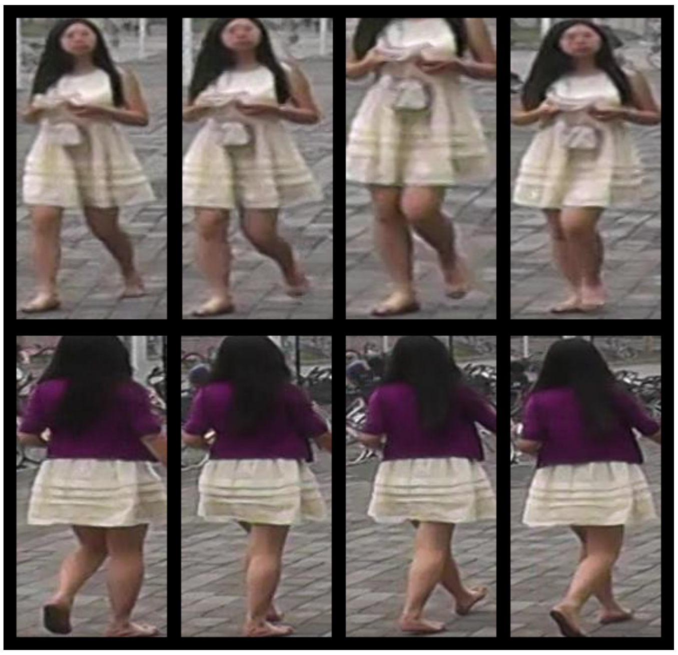


Figure 3.5: Two sequences from the MARS dataset [135] depicting the same person. The change of clothes and high difference in camera angles makes this a challenging re-identification case. The image second to the right in the upper row illustrates how automatic person detectors can result in inaccurate BBs.

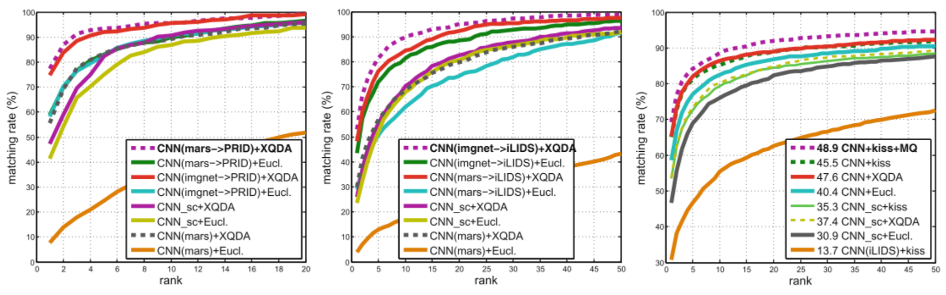
Figure 3.4, MARS includes distractor sequences, which makes the dataset more realistic because distractors are bound to be found in any real-world application.

Although the height of the mounted cameras is not reported by the authors, it seems to be slightly above head-height, which is similar to the perspective of a vision sensor mounted on a typical mobile robot. The fact that there are six different cameras recording at different points in time results in high variation in viewpoint and lighting conditions. Additionally, people are performing different activities (e.g. walking, running, and bicycling) and some people even change clothing across sequences (e.g. wearing t-shirt in one sequence and a sweater or jacket in another sequence). An example of two challenging re-ID video sequences can be seen in Figure 3.5.

3.2.2 Generalizability

Along with the release of MARS, the authors trained a CNN with three different distance metrics on their dataset and evaluated the models on two other well-known video-based re-ID datasets, PRID-2011 [41] and iLIDS-VID [114]. They report the cumulative matching characteristics (CMC) curves (see Section 5.2.1), and show that the model performs relatively well on a completely separate dataset. When fine-tuning the weights of the CNN using the target datasets, the performance is near the state-of-the-art. The results as reported in their paper can be seen in Figure 3.6.

A robotic re-ID algorithm trained on a dataset that do not realistically depict real-world data is likely to generalize poorly when deployed on a mobile robot. The aforementioned observations do, however, indicate that the MARS dataset may have a smaller reality gap compared to other re-ID datasets. This may mean that a re-ID model trained on MARS can yield good performance when introduced to new, unseen data, i.e. video captured from a mobile robot. This dataset attribute is important for this project as the goal is to create a re-ID system that can be deployed on a mobile robot and perform well on real-world data.



(a) PRID-2011 (b) iLIDS-VID (c) MARS

Figure 3.6: CMC curves (see Section 5.2.1) on three video re-ID datasets. In (a) and (b), "CNN(mars)" and "CNN(mars→PRID/iLIDS)" show the model performance of the respective datasets when the CNN is trained solely on MARS and when the CNN is trained on MARS and fine-tuned on PRID-2011 or iLIDS-VID, respectively. This figure is from the MARS paper [135].

3.2.3 Unexplored Aspects

Although numerous re-ID research has been conducted on the MARS dataset since its release [3, 20, 71, 72, 100, 120, 121, 127], these projects mainly focus on improving benchmark results, which are measured by CMC and mAP (see Section 5.2.1), or exploring the use of semi-supervised or fully unsupervised learning methods to reduce the need of labeled datasets. These benchmark evaluations are done in a closed-world environment where all probe persons are known in advance, which, as discussed in Chapter 2, is an unrealistic assumption in a robotics context.

Additionally, algorithm efficiency is rarely prioritized or reported in existing re-ID research. While a surveillance application, which is the main target application for re-ID systems, may not necessarily always require high efficiency, mobile robots rely on algorithms that run in real-time in order to operate smoothly. Algorithm efficiency is therefor a crucial aspect of any re-ID system for mobile robots.

In light of these observations, the MARS dataset is used to evaluate the developed model without assuming that every probe identity is known in advance (Section 5.3.3). Furthermore, the model efficiency is evaluated in terms of FPS, showing that it is possible to achieve benchmark results comparable to state-of-the-art re-ID systems while preserving the high frame rate requirement in the field of robotics (Section 5.3.4).

Chapter 4

Real-Time Person Re-identification System Using a Siamese Neural Network

This chapter presents an in-depth description of the essential modules in the proposed person re-ID pipeline. Various aspects regarding details of the system architecture are discussed along with some encountered challenges. This chapter is divided into three sections: Section 4.1 briefly presents the overall re-ID pipeline, Section 4.2 describes how the feature extraction is performed, and Section 4.3 discusses how the raw features are used to determine a person's identity.

4.1 System Overview

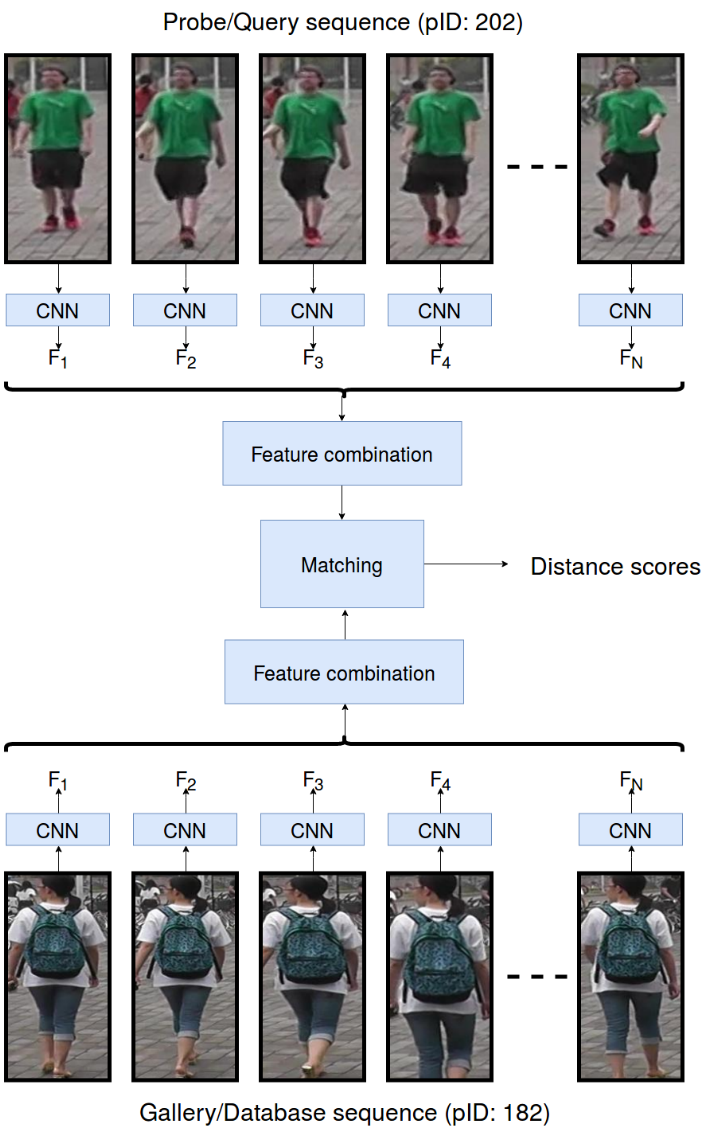


Figure 4.1: The proposed re-ID pipeline consisting of a Siamese CNN. Note that this figure only depicts one gallery person and one probe person, while in reality there are multiple probe and gallery persons.

The objective of the re-ID system is to assign a person identity, which is referred to as a person ID (pID), to all probe, or query, persons (see Background Section 2.3.1 for a definition of a probe person and other re-ID terminology). Robotics requirements are taken into account when designing the re-ID pipeline to ensure that the model developed in this work is suitable for a mobile robotics application. These requirements include that the model can tackle an arbitrary, and potentially large, amount of different probe and gallery people, without having any preliminary knowledge about these individuals. The model also needs to be robust against changes in camera viewpoint, varying lighting conditions, and changes in human pose and orientation. Most importantly, the re-ID system needs to be as lightweight as possible and run at a high frame rate to ensure good performance on low-cost hardware components and to operate efficiently in real-time.

Figure 4.1 depicts the proposed re-ID pipeline consisting of a Siamese CNN. First, features are extracted from the query and gallery images. Secondly, a feature combination module combines all features of a video sequence. Finally, a matching module computes a distance score between the feature representing the probe video and all features representing the gallery videos. The probe person is assigned the identity corresponding to the identity of the gallery person with the smallest distance score (most similarity).

In order to predict the pID of every probe person, the proposed re-ID system relies on two essential components, namely a feature extraction component and a feature matching component. The objective of feature extraction is to find and extract discriminative features from the input data. These features are used to create person descriptors of the probe person and of all gallery persons. Then, once the person descriptors have been created, the feature matching component compares the person descriptor of the probe person against the person descriptors of each person in the gallery, computing a numeric distance for each comparison. A shorter distance means higher similarity, while a longer distance represents higher dissimilarity. Finally, the pID of the gallery person with the highest similarity to the probe person is assigned as the probe person's ID.

4.2 Feature Extraction

Figure 4.2 depicts how the network extracts a feature vector from one video frame. Following the recent trends in re-ID research, and the CNNs superior ability to extract features from image data, the developed re-ID system uses a CNN to learn and extract discriminative and useful features. The CNN processes one video frame at a time and produces a 128-dimensional feature vector from each frame. Each video will therefore have N 128-dimensional feature vector, N being the length (number of frames) of a video.

4.2.1 Network Architecture

The majority of state-of-the-art CNN-based feature extractors in re-ID are based on very deep CNNs such as AlexNet [57], VGG-16 [96], GoogLeNet [101], or ResNet [38]. These networks are trained on large-scale image datasets and thereafter fine-tuned on re-ID datasets. While this approach yields good results on dataset benchmarks, it has two major drawbacks. First, training very deep CNNs is time-consuming and requires a larger amount of training data compared to shallower CNNs with fewer learnable

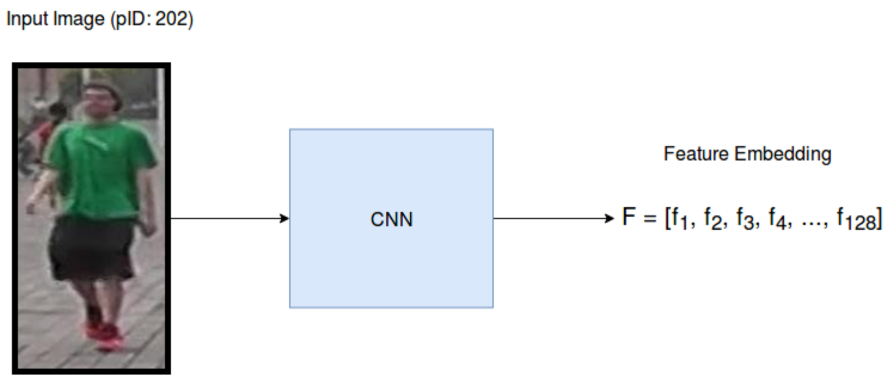


Figure 4.2: The CNN outputs a feature vector, or feature embedding, (F) with 128 float values that together represent the characteristics of the input image. The feature vectors are later compared to match similar people.

parameters. It is therefore common that very deep CNNs are trained on large-scale datasets, such as ImageNet [59], before they are trained for a specific application, such as person re-ID. Second, the deeper the CNN, the more memory and time is required to process the data when the model is deployed on a real application. Both memory and time are limited and critical resources in robotics, and it is therefore necessary to keep CNNs as lightweight as possible.

Based on this observation roboticists typically develop shallow and lightweight CNNs to better fit the needs of mobile robotics. Hermans et al. [40] recently developed a shallow Siamese CNN, which they coined *LuNet*, for re-ID. Despite being very lightweight, their network achieved near state-of-the-art results on the Market-1501 dataset [137] and the MARS dataset [135] . However, the authors did not report any results in terms of processing speed (frame rate), which is crucial for robots that require real-time performance.

This thesis considers three different network architectures. Inspired by the performance of LuNet [40], the LuNet network architecture was reproduced. Next, it is experimented with developing shallower and more lightweight architectures that are more suitable for robotics. To that end, two other networks, *LuNet Light* and *LuNet Lightest* are implemented. LuNet Light contains three fewer layers than LuNet. LuNet Lightest contains five fewer layers than LuNet, and is the most lightweight network presented in this thesis. Table 4.1 provides an overview of the three network architectures.

LuNet Light and LuNet Lightest were built using architectures similar to that of LuNet. There are two reasons to experiment with shallower architectures. First, shallower networks are generally more efficient, which means they are more suitable for robotics. Second, manipulating the number of layers affects the number of learnable parameters. More learnable

LuNet [40]	LuNet Light	LuNet Lightest
conv #1	conv #1	conv #1
res #1	res #1	res #1
pool #1	pool #1	pool #1
res #2	res #2	-
res #3	res #3	-
res #4	res #4	res #4
pool #2	pool #2	pool #2
res #5	-	-
res #6	-	res #6
pool #3	-	pool #3
res #7	res #7	-
res #8	res #8	res #8
res #9	res #9	res #9
pool #4	pool #4	pool #4
res #10	res #10	res #10
res #11	res #11	-
pool #5	pool #5	pool #5
res #12	res #12	res #12
FC #1	FC #1	FC #1
batch-norm #1	batch-norm #1	batch-norm #1
LReLU #1	LReLU #1	LReLU #1
FC #2	FC #2	FC #2

Table 4.1: The architecture of the three networks that are considered for this project, LuNet, LuNet Light, and LuNet Lightest. A hyphen (-) indicates that this layer exists in LuNet but has been removed in the respective network. "res" is a residual block [38], "pool" is a max-pooling layer, "conv" is a single convolutional layer, "FC" is a fully connected layer, and "LReLU" is the leaky rectified linear unit activation function [75]. The uppermost row is the first layers and the last row is the final output layer.

parameters generally means that the network can learn more complex problems. However, this also requires more system memory and CPU or GPU resources, and it is therefore desirable to keep the number of learnable parameters to a minimum.

For the most part, LuNet consists of two or three residual blocks followed by a pooling layer. In LuNet Light, the same structure is kept, but with one less "*res* $\rightarrow$ *res* $\rightarrow$ *pool*" combination. Since this change is rather small, LuNet Light is expected to perform comparably to LuNet. However, since LuNet Light consists of fewer layers, it should be somewhat more efficient than LuNet. In LuNet Lightest, a different approach is taken by only reducing the number of residual blocks and keeping all pooling layers. Since pooling layers reduce data dimensionality, thus also reducing the number of learnable parameters, it is expected that LuNet Lightest is likely to be the most efficient of the three. However, chances are that removing four residual blocks may have a negative effect on the network performance in terms of accuracy.

Table 4.2 provides an overview of the number of learnable parameters in each network architecture. Somewhat surprisingly, LuNet Light contains significantly more learnable parameters than LuNet and LuNet Lightest, despite having fewer layers than LuNet. This is due to the removal of one pooling layer (pool #3), which affects the data dimensionality and number of learnable parameters in all consecutive layers. LuNet Lightest has 16% fewer learnable parameters than the original LuNet, which is a significant reduction for any light-weight application.

LuNet [40]	LuNet Light	LuNet Lightest
2 897 158	4 329 222	2 440 710

Table 4.2: The number of learnable parameters in each network.

As can be seen in Table 4.1, all three architectures consist of six essential building blocks, namely convolutional layers, FC layers, *Batch Normalization*, activation functions, pooling layers, and *residual blocks*.

Convolutional Layers

All LuNets commence with one convolutional layer (conv #1). This layer takes RGB images with three channels as the input and performs 128 convolutions with kernels of size [7×7]. This layer serves as an initial step to reduce the width and height of the data, while increasing the depth from 3 channels to 128 channels. The first layer typically learns to extract low-level features such as edges, corners, and lines (see Section 5.1.3 in the Experiments chapter for visualization of the activation maps).

Recent successful CNNs commonly increase the number of kernels in each convolutional layer while decreasing the kernel size throughout the network [38, 57, 96, 101]. This technique is applied to all three versions of LuNet. The depth is increased from three channels in the input image to 512 channels in the final convolution, and the width and height is decreased from 64×128 in the input image to 4×2 after the final convolution. An overview over the number of kernels in each convolutional layer may be seen in Table 8.1 in the appendix. In all versions of LuNet, all convolutional layers, except for res #1, are found inside the residual blocks which will be discussed shortly.

Fully Connected Layers

Two FC layers (FC #1 and FC #2) are applied at the very end of all LuNets. There are mainly two reasons why it is useful to include FC layers at the end of all LuNets. First, since the layers are fully connected, they can merge features that have been extracted in local areas by the convolutional layers. This enables the networks to mix visual features from the entire human body when creating the final feature vector.

Second, FC layers can be used to efficiently reduce the feature dimensionality to a small, one-dimensional vector. For instance, the first FC layer (FC #1) reduces the dimensionality from 1 024 to 512 dimension. FC #2 further reduces it to 128 dimensions, producing the output feature vector that is eventually used to compare and match person identities.

Activation Functions

Activation functions are commonly applied right after the convolutional and FC layers, and are needed to introduce non-linearity to the network. Without activation functions, the network would simply perform a linear transformation between input and output, which would highly restrict the network's learning ability, especially in complex problems.

Many different activation functions exist. Factors such as data type, network architecture, and the nature of the problem to solve (e.g. regression, classification, verification etc.) should be taken into account to select an appropriate activation function. In all versions of LuNet, all convolutional layers along with the first FC layer (FC #1 in Table 4.1) are followed by a leaky rectified linear unit [75] activation function, commonly referred to as *leaky ReLU* or *LReLU*. The final FC layer (FC #2 in Table 4.1) is linear and is not followed by any activation function.

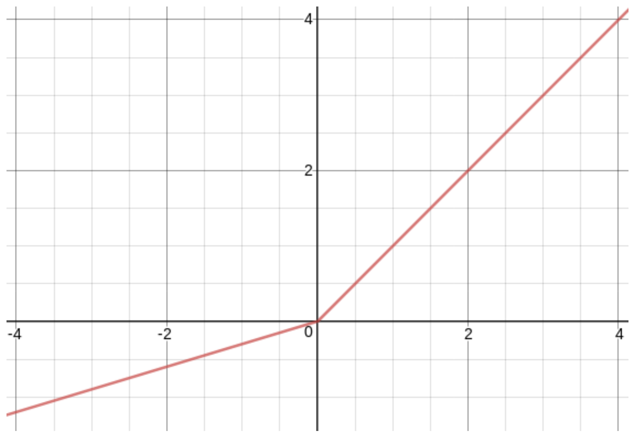


Figure 4.3: Plot of the leaky ReLU activation function with the negative slope constant $\alpha = 0.3$. The x-axis is the value before the LReLU operation and the plotted line, $LReLU(x)$, is the activated value.

Leaky ReLU is a popular activation function because it enables the network to converge faster during training compared to other activation functions, such as the *sigmoid* or the *tanh* activation functions [47]. Leaky ReLU is also very simple to implement and does not involve any heavy mathematical operations, meaning it adds a minimal CPU processing overhead on the network. Furthermore, leaky ReLU does not suffer from the *vanishing or exploding gradient problem*, a phenomenon that, in worst case, can leave the network unable to learn.

Leaky ReLU is defined as follows:

$$LReLU(x) = max(\alpha x, x) = \begin{cases} \alpha x & if\ x < 0 \\ x & if\ x \geq 0 \end{cases} \tag{4.1}$$

where x is the value in any given neuron and α is a hyperparameter that decides the slope of the activation function when $x < 0$ (see Figure 4.3 for a plot of the activation function). Following the recommendations of [40], $\alpha = 0.3$ was used in all versions of LuNet.

Pooling Layers

To reduce the spatial size of the data (width and height), pooling layers are inserted periodically between the residual blocks. More specifically, all LuNets leverage the max-pooling operation to control the data volume. The pooling operation is always performed after convolutional layers or residual blocks, and operates on each depth of the activation maps individually. Both LuNet and LuNet Lightest have five max-pooling layers, while LuNet Light has four max-pooling layers.

In all versions of LuNet, all pooling layers have a kernel size of 3×3 and a *stride* of 2. This means that for each 3×3 patch in the input volume, the highest number is kept and the rest are discarded. Stride of 2 means that the filter slides by a step of 2 pixels before repeating the operation. Figure 4.4 illustrates the max-pooling operation on a 5×5 volume using the same kernel size, 3×3, and stride, 2, as the pooling layers in all LuNets.

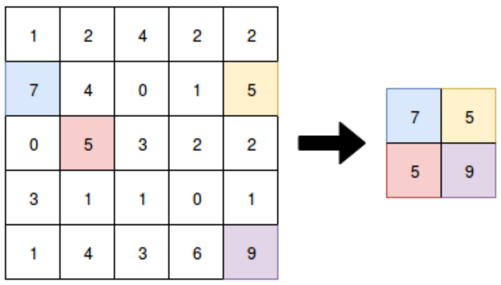


Figure 4.4: Max-pool operation with stride = 2 and a 3×3 filter sliding over a 5×5 activation map (left), resulting in a 2×2 output volume (right). Best seen in color.

Batch Normalization

Batch Normalization [42] is an important algorithm that is commonly used in deep neural networks. Batch Normalization allows for training of deeper network architectures, reduces training time, and makes the model less sensitive about small changes in hyperparameter values [42].

Batch Normalization works by reducing the *covariance shift* in each network layer. Covariance shift is the change in the distribution of the input data. This effect is problematic because each layer in the network continuously has to adapt to the new data distribution. Even a small change in the input data's distribution can negatively affect the learning capability of NNs, especially those networks that are very deep. This is because as the data passes through the different layers and operations, such as convolutions and pooling operations, the potentially small change in the input data distribution is amplified in every layer. If no measures are taken to counteract the shift in data distribution, the network will probably train slowly and learn a sub-optimal function.

To reduce the covariance shift, Batch Normalization normalizes the data in each layer of the network. During training, Batch Normalization normalizes the batch mean (μ_B) and the batch variance (σ_B^2) of each node in the activation maps. μ_B and σ_B^2 are estimated during training, and these estimated values are used to normalize the data during inference time. Since the mathematical operations that Batch Normalization performs differ during training and testing, it is crucial to specify when the network is training and when it is being tested to ensure that Batch Normalization works correctly. In all versions of LuNet, Batch Normalization is applied subsequent too every convolutional operation, including those inside each residual block.

Residual Blocks

Generally speaking, deep networks are capable of learning more complex functions than shallower networks. However, as networks get very deep, they are unable to learn simple functions, and the performance will saturate; furthermore, at some point it will start degrading (known as the *degradation problem*) [38].

Residual blocks were designed to overcome the degradation problem. The residual block was initially proposed by He et al. [38] in a 2016 paper where the authors presented ResNet, a very deep CNN for image classification. The authors designed a CNN consisting of many consecutive residual blocks instead of naively stacking many convolutional layers to increase the network depth. Figure 4.5 depicts the difference between a regular CNN architecture and an architecture consisting of residual blocks.

One residual block is a stack of two or three convolutional layers with an activation function after each layer. The difference between a residual block and a regular stack of convolutions, however, is that residual blocks contain a *skip connection*, or *identity shortcut*, which is a connection from the start of the residual block that is added at the end of the residual block (see Figure 4.6 for a visualization of the skip connection in one residual block). In a regular series of convolutional layers, each layer attempts to learn the ground truth value. In a residual block, however, each block tries to learn the residual between the input and the ground truth. He et al. [38]

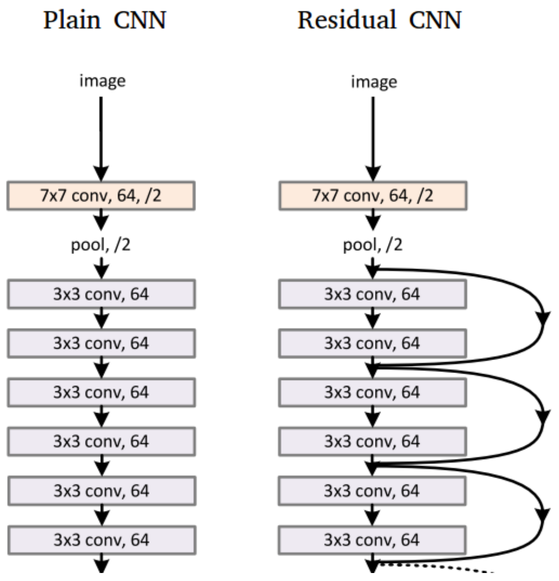


Figure 4.5: The difference between a residual CNN and a plain CNN. The curved arrows in the residual CNN are "skip connections". 3×3 and 7×7 are filter sizes, while 64 indicates the number of filter banks (depth). This figure is a slightly modified excerpt from [38].

observe that it is easier to learn the residual of $output - input$ than directly learning the output. Furthermore, the skip connection helps overcoming the vanishing or exploding gradient problem, allowing for faster learning in deeper networks.

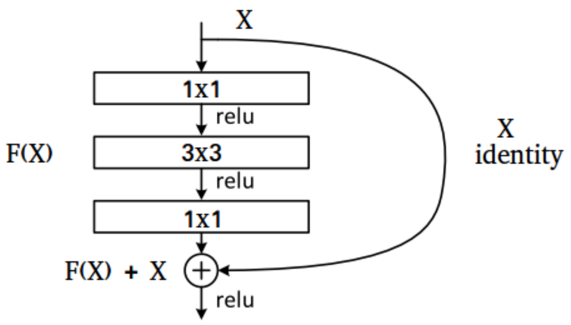


Figure 4.6: A "bottleneck" residual block from [38].

All residual blocks in the three LuNets are of the "bottleneck" type as depicted in Figure 4.6. Bottleneck residual blocks are built up from n_a 1×1 convolutions, followed by n_b 3×3 convolutions, and finally n_c 1×1 convolutions. The first 1×1 convolution in the bottleneck residual blocks reduces the data dimensionality, meaning that the following 3×3 convolution can be performed more efficiently. The efficiency aspect of the bottleneck residual blocks leaves it as an excellent option for networks that need to process the data fast. After the 3×3 convolution there is a final set of 1×1 convolution that normally restores the dimensionality from before the residual block. In some cases, the last convolution increases the depth of the data. In that case, the depth of the identity, X, is increased to match the new dimensionality to make the final addition $F(x) + X$ possible. The dimensionality increase is done by using 1×1 convolutions.

The parameters $n_a, n_b and n_c$ regulate the depth of the activation maps. In the first residual block, the activation maps have 128 channels. This depth is periodically increased, first to 256 channels and finally to 512 channels before the first FC layer (FC #1). Table 8.1 in the appendix specifies the values of $n_a, n_b and n_c$ in each residual block.

Weight Initialization

All kernel values in the convolutional layers and the weights in the FC layers need to be assigned an initial value. The variance scaled initialization [39] is commonly used with the LReLU activation function, and was used to initialize all kernels in the convolutional layers. For both FC layers, *Xavier initialization* [30] was used.

4.2.2 The Triplet Loss

By comparing predictions by the CNN with the ground truth, the loss function tells the network how well it performed. A loss function is essential to train any CNN, and is arguably one of the most important design aspects to consider in order to design successful network architectures.

The loss value is typically calculated once for each *batch*. One batch is the collection of all images that are fed through the network before the loss value is computed. After the loss value is calculated, the optimization algorithm (see Section 4.2.3) updates the learnable parameters before a new batch is fed through the network.

Many loss functions exist for binary and multi-class classification problems. However, as discussed in the background chapter (see Section 2.3.4), classification networks can be problematic in the field of re-ID due to a large amount of classes (pIDs) that differ between the training and test set (the pIDs in the training set are not the same as the pIDs in the test set). To avoid these complications, all three LuNets are Siamese CNNs instead of traditional classification CNNs. Siamese CNNs differ from traditional CNNs in that instead of learning a mapping from an input image to a class, the Siamese networks learn a feature vector, or feature embedding for each input, and tries to predict whether two feature embeddings depict the same person or not. A Siamese network structure is better suitable for a robotic re-ID system because it removes the need of a distance metric learning function, and it allows us to easily and efficiently handle a possibly large and arbitrary amount of different people identities.

To allow the Siamese CNN to learn similarities between input images, the *batch hard triplet loss* proposed by Hermans et al. [40] is used in all versions of LuNet. The triplet loss function first became popular after Schroff et al. [93] successfully used it in their neural network for face recognition, coined *FaceNet*, that improved state-of-the-art results by a large margin. Hermans et al. [40] modified the original triplet loss and achieved impressive results when they applied it to LuNet and the re-ID problem.

In the batch hard triplet loss, each loss value is calculated using the distances between the feature embeddings of three input images, known as *triplets*. These triplets consist of an *anchor*, a *positive instance*, and a *negative instance*. The anchor and the positive instance are feature embeddings that represent the same identity (two images of the same person), while the negative instance is a feature embedding representing a dissimilar identity (the image does not depict the same person as the anchor). The batch hard triplet loss is defined as

$$L(\theta; X) = \sum_{i=1}^{P}\sum_{a=1}^{K}\Bigg[\ln\Bigg(1+\exp\Bigg(\overbrace{\max_{p=1\rightarrow K}\left(f_\theta(x_a^i), f_\theta(x_p^i)\right)}^{\text{hardest positive}} - \underbrace{\min_{\substack{j=1\rightarrow P\\ n=1\rightarrow K\\ j\neq i}}\left(f_\theta(x_a^i), f_\theta(x_n^j)\right)}_{\text{hardest negative}}\Bigg)\Bigg)\Bigg]_+ \tag{4.2}$$

where X is the collection of all images in a batch, θ is the collection of all network parameters, P is the number of different identities in the batch, K is the number of images of each identity, $f_\theta(x)$ is the feature vector of one image as produced by LuNet (θ) and *max* and *min* are functions that respectively return the maximum and minimum distance between the two feature vectors. Following the recommendations by Hermans et al. [40], the Euclidean distance is used to measure the distance between feature vectors.

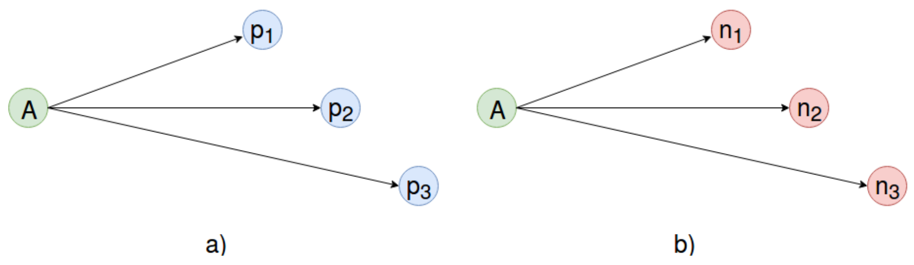


Figure 4.7: a) Anchor with three positive samples. b) Anchor with three negative samples. The hardest positive is the sample with the same identity as the anchor (A), but with least similarity (p_3 in the figure). The hardest negative is the sample with a different identity than the anchor (A) that has the highest similarity (n_1 in the figure). Longer arrow represents higher distance (dissimilarity) between the feature vectors. Best seen in color.

For each image, or anchor, in a batch, the first term in Equation 4.2 ($max(a, b)$) returns the *hardest positive* and the second term ($min(a, b)$) returns the *hardest negative*. Of all possible image pairs for each image in a batch, the hardest positive is the image pair that depicts the same person, but also has the greatest measure of dissimilarity (i.e., largest distance, see Figure 4.8). Similarly, out of all possible image pairs for each image in a batch, the hardest negative is the image pair that depicts different persons but has the highest measure of similarity (i.e., shortest distance, see Figure 4.9). In other words, hard positives are image pairs of the same person that are hard for the network, possibly because of high variance in viewpoint, challenging lighting conditions, or occlusions. Hard negatives are image pairs of different people that the network believe depict the same person, possibly due to similar clothing style or clothing color. Figure 4.7 provides a visualization of hard positives and hard negatives represented in a two-dimensional space.

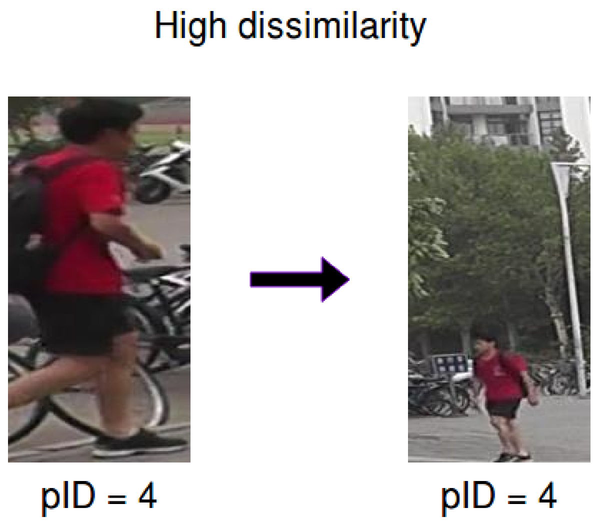


Figure 4.8: An example of a hard positive image pair. Both images depict the same person, but a largely inaccurate BB in the right image along with with a highly different camera angle makes difficult to tell that both images depict the same person.

Figure 4.10a) depicts a hard triplet with a high loss value, and Figure 4.10b) depicts an easier triplet with a smaller loss value. As the network is presented with many hard triplets during training, it learns to push the negative instance further apart from the anchor and at the same time pulling the positive instance closer to the anchor, i.e. more triplets like Figure 4.10b) and less triplets like 4.10a).

Hard triplets can cause the training to be very slow, or even stagnate early in the training phase if the triplets are too hard for the network to learn.

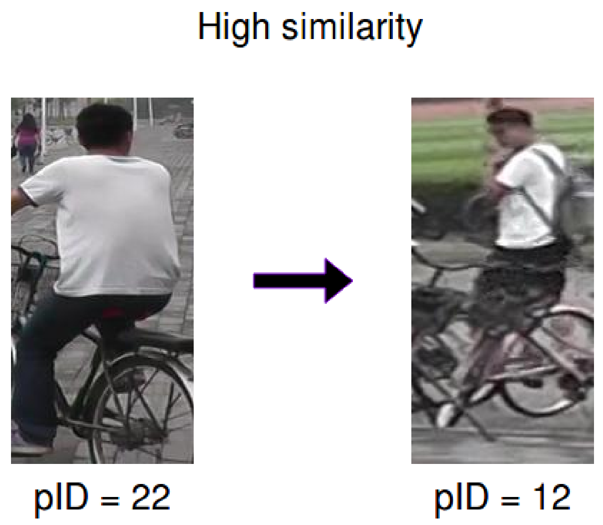


Figure 4.9: An example of a hard negative image pair. The images depict different persons, but similarity in terms of clothing, hair color, and camera angle, along with and the presence of bikes in both images, makes it challenging, even to the human eye, to tell that the two images do not depict the same person.

Shallower networks are especially prone to this issue because they cannot learn as complex functions as deeper networks. Hermans et al. [40] did not encounter this issue with LuNet, however, it may become problematic for networks that have less convolutional layers. In light of this, the developed system avoids this potential problem by removing constraints regarding which images are selected when creating the batches. This means that there may be two images of the same person from the same video. Two images from the same video are likely to have less variance than images from different videos, because there is a high similarity in camera angle, clothing color, and light conditions. Intuitively, this should yield slightly easier triplets, which again reduces the chance of having little or no progress early in the training phase. This differs from the implementation of Hermans et al. [40], which selected images from different videos.

Batch Size for the Triplet Loss

To ensure proper functioning of the batch hard triplet loss, it is important to carefully select the size of the batches and which pIDs are included in each batch [40]. Each batch (B) needs to contain K images of P different persons, resulting in a batch size of $B = P \cdot K$. Hence, every batch contains $P \cdot K$ unique triplets, where each image in the batch is used as an anchor once, but may be used as the hard positive or hard negative either zero or multiple times in the other triplets. The batch hard loss value is then

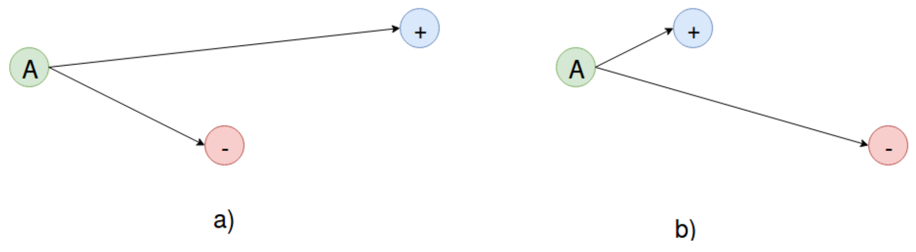


Figure 4.10: Visualization of a hard triplet where the anchor is closer to the negative instance (a) and an easier case where the anchor is closer to the positive instance (b). In re-ID, a positive instance (blue circle with a "+") is a person with the same ID as the anchor (green circle with "A"), while the negative instance (red circle with a "-") is an other person. Shorter arrows (distance) represent more similarity. Best seen in color.

calculated on each triplet, such that the loss function is made up by the sum of a total of $P \cdot K$ float values. P and K are hyperparameters that are more closely discussed in Section 5.1.1.

4.2.3 Optimization Algorithm

As briefly discussed in background Section 2.3.3, optimization algorithms are commonly used to train CNNs. The optimization algorithm is a mathematical function whose goal is to find the combination of learnable parameters that minimize the loss function. In CNNs, the filters in the convolutional layers and the weights in the FC layers are the learnable parameters, and in all LuNets the batch hard triplet loss function [40] (Section 4.2.2) is the loss that the optimization algorithm aims at minimizing.

The *Adam* optimization algorithm [51] was used to optimize the network parameters in all three versions of LuNet. Adam is a first-order optimization algorithm, meaning that it uses the gradient values of the loss function with respect to the trainable parameters to update the network parameters in a manner that minimizes the triplet loss. It has been shown to be excel to train networks that have many trainable parameters and when the network is trained on large datasets [51]. Furthermore, Adam was used by Hermans et al. [40] to train LuNet, where the authors showed that it is well-suited to optimize Siamese network architectures and the triplet loss.

As with most optimization algorithms, Adam requires a set of specified hyperparameters: 1) the learning rate α_{lr} (not to be confused with the slope parameter α in the leaky ReLU function), 2) the decay rates β_1 and β_2, and 3) a numeric stability constant ϵ. The learning rate decides how quickly the network learns, i.e. how much the weights are updated in each backward pass. The decay rates are used to control the gradients' moving average. The numeric stability constant is needed simply to avoid division by zero.

Values for the decay rates and the numeric stability constant are less critical than the learning rate, and the authors' recommended default values [51] are therefore used. The specific hyperparameter values used for training are are discussed in Section 5.1.1.

4.2.4 Data Preprocessing

Before the network can be implemented and trained, the data needs to be preprocessed. The *data preprocessing* is a composition various operations that are applied to the dataset before the data is used to train the model.

The first aspect to consider when it comes to data preprocessing is the dataset size. The MARS data set, with its almost 1.2 million images that necessitates 6.8 GB of storage space, requires some careful considerations when it comes to how the data is processed to ensure efficient training. When feeding the dataset to the network, the easiest option is to load the entire dataset to memory before starting the training process. While this is a simple and feasible approach for small datasets, the size of MARS and the hardware available for this thesis (see appendix Section 8.1 for hardware details) makes it unsuitable for this work. Instead of loading the entire dataset to memory at once, only parts of the dataset are loaded at a time. When a new batch is needed, only the images of P different pIDs are loaded to memory, leaving enough memory available to train the network.

Loading the dataset to memory is a time-consuming task, especially when the data consists of larger files, such as images. If the dataset is loaded to memory all at once, this time-consuming task is only performed once, and is therefore less critical. When the dataset is loaded in an on-line manner when the data is needed, however, this process is repeated frequently and can become the bottleneck during training. To minimize this bottleneck, the dataset is converted to *Tensorflow Records*[1], or *TFRecords*, which is a simple file format to store sequences of binary files, or records. It was found that converting the MARS dataset to TFRecords sped up the LuNet training time by at least a factor of ten.

In DL, it is common and good practice to shuffle the input data before training the network. Randomizing the order of the training data ensures that the network is presented with a greater variety of classes, or pIDs, in each batch, making it easier to minimize the loss function. When the entire dataset is stored in memory, it can easily be shuffled with one simple line of code. When only a portion of the dataset is stored in memory at a time, however, shuffling this portion will only randomize a potentially small subset of the data. In many cases, this will lead to very poor randomization, for instance if most of the data loaded is from the same class, and is therefore not an applicable approach for this thesis. Another possibility is to shuffle the entire dataset and store the shuffled version on disk. This is however a time-consuming operation that should be avoided when possible.

[1]https://www.tensorflow.org/tutorials/load_data/tf_records

To overcome issues related to randomizing the data, shuffling is performed in two steps. The first step is to randomly sample P pIDs (remember that a pID is simply an integer that represent one individual, not the actual images or videos of this individual) and load all images corresponding to these pIDs to memory. Instead of having all images of the 625 persons in the MARS dataset loaded in memory, only the subset of P images that are needed are stored in memory at any given time. The second step is to randomly select K images of each of the pIDs currently stored in memory. Since this is a very small subset of the training data, it can easily be shuffled in memory without any operations that are very time-consuming. This was considered to be the least computationally expensive way to fully randomize the data while keeping the requirement of having K images of P pIDs in each batch.

After loading and randomizing the dataset appropriately, the data is augmented by resizing all images and applying random flips and crops which is common practice when training CNNs [57]. Random crops and flips ensures that the network sees a greater variety of input images, meaning it will be more robust towards inaccurate BBs and changes in camera point of view. Following the procedure by Hermans et al. [40], the images are first converted to a size of $H = 144, W = 77, D = 3$, where H is the height, W is the width and D is the depth. Then, the images are randomly cropped to a size of $H = 128, W = 64, D = 3$. D is kept constant to keep the three color channels in RGB images. The cropping is followed by a 50% chance of horizontally flipping the image. Figure 4.11 illustrates the cropping and flipping operations.

The original size of all MARS images is $H = 256, W = 128, D = 3$, meaning that the cropping has reduced the image size by 75% while keeping the same aspect ratio. Smaller images allow for a larger batch size without running out of memory because passing each image through the network requires less memory.

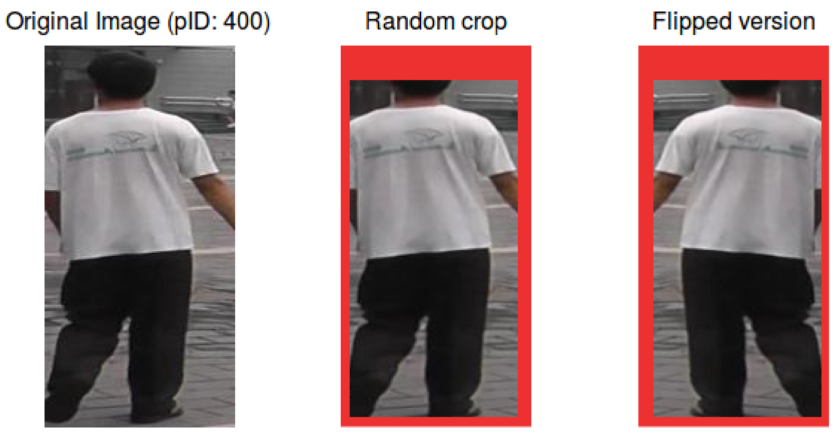


Figure 4.11: Example of one image from the MARS dataset (left), after random crop is applied (middle), and the horizontally flipped version (right). This specific image has been cropped by 10 pixels from the left, 6 pixels from the right, 25 pixels from the top, and 7 pixels from the bottom. The red area marks the image portion that has been cropped away. Horizontal flip is applied after the crop, which is why the cropped margins at the left and right hand side are swapped.

4.3 Feature Matching

4.3.1 From Feature Embeddings to Person Identities

This far the method used to extract the 128-dimensional feature vectors from each person image has been discussed. What a re-ID system is tasked at finding, however, is the actual identities of the persons depicted in the probe images. To achieve this, a function that compares the feature vectors of people with an unknown pID (F_{known}) against feature vectors of people with a known pIDs ($F_{unknown}$) is needed. As can be seen in Figure 4.12, all gallery persons have a unique identity number and are therefore in the category F_{known}, whereas the identity of all probe persons is unknown, $F_{unknown}$, and to be determined by the re-ID system. The objective of the feature matching component is to compare the feature vectors of all the probe persons against the feature vectors of all gallery persons, and assign the correct identity number to all the probes.

Figure 4.13 depicts the feature matching component in the re-ID pipeline. The matching component requires two inputs, namely the feature vector that represents the probe person, and the feature vectors and pIDs of all gallery persons. The feature vectors of all gallery persons are extracted in advance and stored in a database, while the feature vector of the probes need to be extracted in an online manner as the robot perceives people.

The output of the matching component is a vector containing the

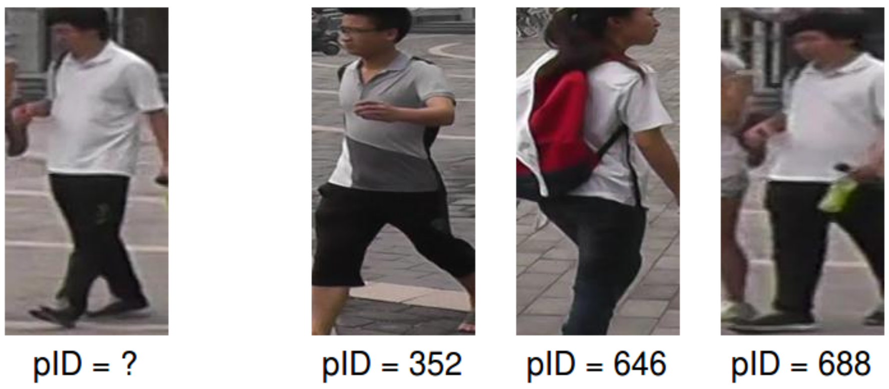


Figure 4.12: Example images extracted from videos in MARS. Left: one probe person with an unknown identity ($F_{unknown}$). Right: three gallery people with known identity (F_{known}).

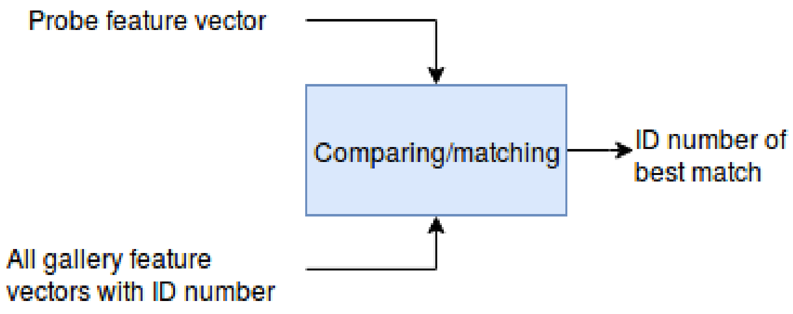


Figure 4.13: The feature matching function compares the probe feature vector against all gallery feature vectors and assigns an ID based on the ID of the best match.

identification number of all gallery persons sorted according to their similarity with the probe person. For instance, if the matching component outputs the result $R = [5, 8, 5, 1, 4, 2]$, the gallery person with ID number = 5 (first/leftmost position in R) has the highest similarity to the probe person. Furthermore, there are 5 unique numbers in R, meaning that there is a total of five different person identities in the gallery. Note that person ID = 5 is found twice in R. This means that the gallery contains feature vectors representing two different videos of the same person. Variables such as varying camera viewpoint, challenging lighting conditions, and change of clothes are plausible reasons why person ID = 8 has a higher resemblance to the probe person than one of the feature representation of person ID = 9.

4.3.2 Combining Features

Section 4.2 only discussed how to extract the feature vectors for each image individually. In this thesis, however, re-ID is considered on the video level, meaning that features need to describe entire video sequences instead of single images. To obtain features that represent video sequences, the individual feature vectors from each image frame in a video need to be combined.

There are several different ways to combine feature vectors. Zheng et al. [135] evaluated three different feature combination methods along with the publication of the MARS dataset. For example, consider a very short video sequence of three image frames with four numeric values per feature vector (in reality the video sequences are longer and of arbitrary length, and the feature vectors produced by LuNet each consist of 128 numeric values, but the idea is the same). Each image frame has been processed by LuNet and converted to a feature vector representation. These feature vectors are: $f_1 = [1, 4, 3, 2]$, $f_2 = [6, 3, 4, 2]$, $f_3 = [3, 5, 9, 0]$, where f_1, f_2, and f_3 each represent one consecutive frame in the video.

This thesis considers three different strategies to combine features. First, only the feature vector of the first frame, f_1 is used to represent the entire video sequence. In our example, the feature vector describing the entire video would then be $F = f_1 = [1, 4, 3, 2]$. Second, the element-wise mean is computed across all feature vectors in the video sequence. Referring back to our example, the resulting feature vector would be $F = (f_1 + f_2 + f_3)/3 = [3.3, 4, 5.3, 1.3]$. Third, the element-wise max-pooling operation is used. This resembles the max-pooling operation described in Figure 4.4, but instead of pooling across the height and width of the activation maps, the pooling operations is done by selecting the highest activation value at each index in the feature vectors. In our example, the max-pooled feature vector would be $F = maxpool(f_1, f_2, f_3) = [6, 5, 9, 2]$. In this thesis, these three three methods of combining features are referred to as *single-image feature*, *mean feature*, and *max-pool feature*, respectively.

A clear difference between combining features by calculating the mean or the max-pool and using the single-image feature is that the latter only considers one image while the mean and max-pooling features consider the entire video sequence. Using single-image features is therefore essentially equivalent to doing image-based re-ID instead of video-based re-ID, meaning that the temporal information found in video sequences is omitted. It is argued that this is suboptimal because temporal data contains rich information that may contribute to extract more unique features, thus improving the re-ID performance. Single-image features are however less computationally expensive to use because the re-ID model only has to process one image, instead of an entire video sequence, to produce the feature vector. A comparative result of the three feature combination methods applied both on the probe and gallery data may be found in Section 5.2.2 in the Experiment chapter.

4.3.3 Distance Metrics

Once the feature vectors have been combined to form representations of video sequences instead of still images, a distance metric is used to compute the similarity of the vectors. In re-ID systems where features are hand-computed or extracted by a classification-style CNN, this block in the re-ID pipeline requires a distance metric learning algorithm (Section 2.3.2) to learn a metric appropriate for the feature space. In the proposed Siamese CNN architecture, however, the batch hard triplet loss [40] function specifically minimizes the Euclidean distance between feature vectors. Instead of applying a distance metric learning step, the Euclidean distance (as defined in Equation 2.2 in the background chapter) is used to assign distances between the probe feature and all gallery features. A short Euclidean distance between two feature vectors mean that the network has produced similar feature vectors for the two inputs, indicating that they are more likely to depict the same person. A long Euclidean distance between two feature vectors mean that the network believe these two people to be highly dissimilar.

Chapter 5

Experiments

This chapter discusses the various experiments conducted on the three network architectures: the reproduced LuNet, LuNet Light, and LuNet Lightest. The three network architectures are first evaluated using the MARS evaluation protocol [135]. Then, the proposed evaluation measures that are more relevant to a robotic re-ID system are presented along with the results obtained by LuNet, LuNet Light, and LuNet Lightest.

5.1 The Training Process

Training deep CNNs is an iterative process that consist of feeding batches through the network, computing the error, or loss, of the given batch, and finally updating the trainable weights according to the loss function. There are multiple aspects that need to be considered before and while training the network, such as hyperparameter selection, variable monitoring, and visualization of the activation maps and embedding space.

5.1.1 Hyperparameters

There are two types of different parameters in CNNs. The first are the learnable parameters, which are mostly found in the convolutional kernels, whose optimal values are learned during training. The second are hyperparameters whose values cannot be learned while training the model, but instead have to be determined in advance. The process of finding a combination of good hyperparameters is known as *hyperparameter tuning*. Selecting correct hyperparameter values is crucial to train any network [47].

The collection of hyperparameters needed to train all versions of LuNet are listed in Table 5.1. Note that all hyperparameters except for the batch size, B, have the same values when training all three networks, allowing for a fair comparison of the performance of each architecture. This section discusses the selection of each hyperparameter value.

	LuNet (reproduced)	LuNet Light	LuNet Lightest
α	10^{-3}	10^{-3}	10^{-3}
α_{lr}	Eq. 5.1	Eq. 5.1	Eq. 5.1
s_{max}	45 000	45 000	45 000
s_{decay}	$0.6s_{max} = 27\,000$	$0.6s_{max} = 27\,000$	$0.6s_{max} = 27\,000$
β_1	0.9	0.9	0.9
β_2	0.999	0.999	0.999
ϵ	10^{-8}	10^{-8}	10^{-8}
$B = P \cdot K$	$B = 18 \cdot 4 = 72$	$B = 18 \cdot 4 = 72$	$B = 20 \cdot 4 = 80$
Crop+flip?	yes	yes	yes

Table 5.1: Hyperparameter values used to train each network. α is the negative slope constant in the leaky ReLU activation function, α_{lr} is the learning rate, s_{max} is the max number of training steps, or iterations, s_{decay} is the iteration where α_{lr} starts decaying, and B is the batch size.

Learning Rate

The learning rate (α_{lr}) is the most important hyperparameter in the optimizer function, regulating how fast the network learns, i.e. how much the trainable parameters are updated in each backward pass. High learning rate reduces training time, but can hinder the network from finding the global minimum. On the other side, a low learning rate restricts the network's ability to overcome local minimum, and requires more time to train. Therefore, there is a tradeoff between a low and high learning rate that need to be taken into consideration for experiments.

To address this tradeoff, a common approach is to have a high learning rate early in the training phase, before reducing it to allow the network to find an optimal minimum. Following the values used in [40], the following dynamic learning rate was used in all experiments:

$$\alpha_{lr}(s) = \begin{cases} 10^{-3} & if\ s < s_{decay} \\ 10^{-3} \cdot 001^{\frac{s-s_{decay}}{s_{max}-s_{decay}}} & if\ s \geq s_{decay} \end{cases} \tag{5.1}$$

where s is the current training step (i.e. number of epochs thus far), s_{decay} is the training step where the learning rate starts decaying, and s_{max} is the final training step. Table 5.1 summarizes values used for s_{decay} and s_{max}.

The Adam optimizer function also requires three additional parameters: two decay rates β_1 and β_2, and a numeric stability constant ϵ. For these parameters, the authors provide recommended default values that they show work well for a large range of optimization problems [51]. To limit the amount of hyperparameters, and thereby reduce the hyperparameter search space, the parameter setting consistent with the recommended default values [51] were used in all experiments: $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$ [51] in all experiments.

Stopping Criteria

The *stopping criteria* decides when to terminate the training process. A common way to decide when to stop training is to split the training data in two, one part for training and one for validation. The network is evaluated on the validation set at a periodic interval while training. Training stops when the accuracy on the validation set starts decreasing. This is known as overfitting the network to the training set; in other words, it refers to the network's ability to generalize to new, unseen data, which becomes challenging to achieve if training continues.

While many classification-style CNNs are very prone to overfitting, none of the training runs in the conducted experiments showed any signs of overfitting. This is because during training, the Siamese architecture combined with the batch hard triplet loss [40] learns to group similar people together in a cluster and simultaneously push different people apart. Training for a longer period will only decrease the distances within each cluster (resulting in shorter distances between similar people) and increase the distances between clusters (resulting in longer distances between dissimilar people). In fact, long training times on Siamese networks that use the batch hard triplet loss [40] does not contribute to overfitting, but can on the contrary have a positive effect on the networks ability to generalize to unseen data due to the more optimized distances within and between clusters.

Since the architectures of all three LuNets are not prone to overfitting, the data was not split into training and validation sets. Instead, the official MARS dataset that consists of 625 pIDs for training and 636 pIDs for testing [135] was used. The entire training set was used to train, and training was simply stopped after a fixed amount of iterations, or steps, (s_{max}). Although no training runs showed the tendency to overfit the training data, experiments with very long training times (s_{max} = 100 000) did not show any significant improvements on the test set. To keep the training times to a minimum, s_{max} = 45 000 was used as this was observed to be the point where the accuracy saturated.

Batch Size

In all experiments, mini-batches of size $B = P \cdot K$, where P is the number of pIDs in a batch, and K is the number of images of each person, were used. Selecting an appropriate batch size is important because it affects the networks learning capability and training time. Mini-batches of sizes between $B = 32$ and $B = 512$ are known to improve the network's ability to generalize to unseen data compared to batches of size $B > 512$ [49]. On the other hand, smaller batches require more training steps (higher s_{max}) to reach the same amount of passes through the network. Finally, larger batches require more memory; therefore, the maximum batch size is limited by the hardware available.

Taking the aforementioned variables into considerations, it was decided to use the largest possible mini-batch size, meaning that the limitations are $B \leq 512$ and *memory needed < total memory available*. For LuNet and LuNet Light, $B = 18 \cdot 4 = 72$ was the largest possible batch size. For LuNet Lightest, whose architecture requires less memory due to fewer learnable parameters, $B = 20 \cdot 4 = 80$ was used.

The batches are selected *without replacement*, which means that the P classes (pIDs) used in each mini batch are selected randomly. While this can give greater variety in the batches, it may result in some pIDs being used more often than others. To ensure that there is not too much variance in the distribution between the pIDs during training, the pIDs used in each batch were tracked and plotted after training. The statistics can be seen in Figure 5.1, where each plotted point shows how often the respective pID has been used in a batch. Since the pIDs are randomly selected, some spread is to be expected. There are no extreme values and the distribution is fairly even, indicating that the network is given a high variety of different combinations of pIDs in the batches during training.

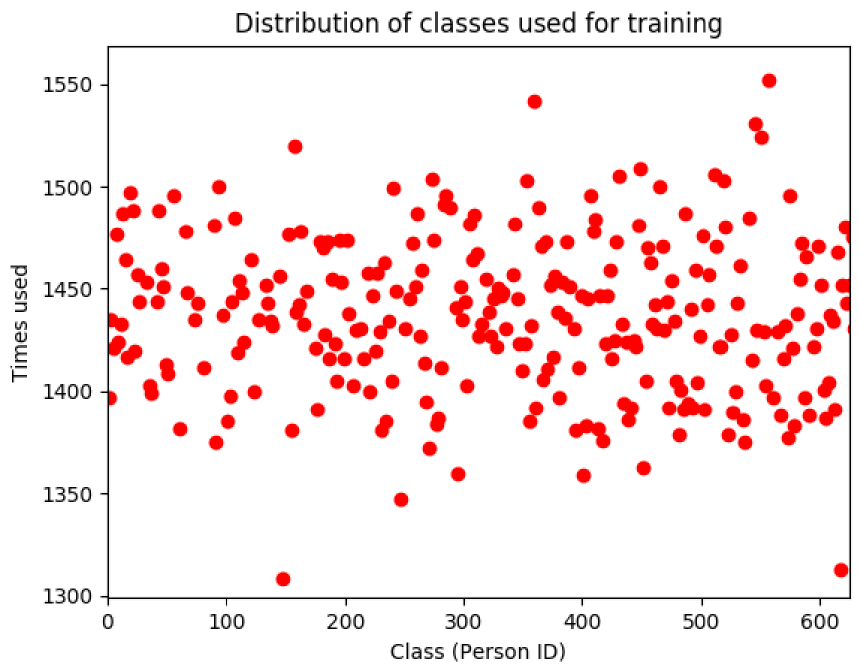


Figure 5.1: Distribution of how often each of the 625 individual persons were used during training of LuNet Light.

5.1.2 Variable Monitoring

Training deep and complex CNNs can be a challenging process. Errors are likely to occur and often difficult to debug without the correct tools. To ease debugging and the process of finding good hyperparameters, it can be useful to plot certain parameters during training. In this thesis, Tensorflow's *tensorboard* [1] was used for the purpose of monitoring variables during training and Matplotlib's *pyplot* [2] library was used to plot statistics.

The loss function is arguably the most useful variable to carefully monitor. A loss that decreases as the network trains is a good indication that the learning process runs correctly. Figures 5.2, 5.4, and 5.3 respectively show how the batch hard triplet loss [40], the train-set accuracy, and the learning rate evolve over time as the network trains. Figure 5.2 shows that the loss value decreases as the network trains, which indicates that the training is progressing as desired. The steepest decrease in the loss happens early in the training phase as the network is introduced to new triplets. After roughly $\frac{1}{3}$ of the training time ($s = 15\,000$), the steepness of the loss curve starts decreasing. This indicates that the network has seen most of the easier triplets, while some harder triplets still need to be learned. Furthermore, the loss is very close to zero once the learning rate starts decaying. In this phase of the training, the learning is slower, meaning that the network fine tunes its

[1] `https://www.tensorflow.org/guide/summaries_and_tensorboard`
[2] `https://matplotlib.org/api/pyplot_api.html`

learnable parameters to reach a local minimum. Along with the decreasing loss function, the accuracy on the training set rapidly increases early in the training before stabilizing right below 100 % accuracy, which also indicates that the network is training correctly. Note that the model accuracy during training is measured on the same data that is used to train, meaning that it does not precisely represent how well the model will perform on new, unseen data, but rather indicates that the training evolves as expected.

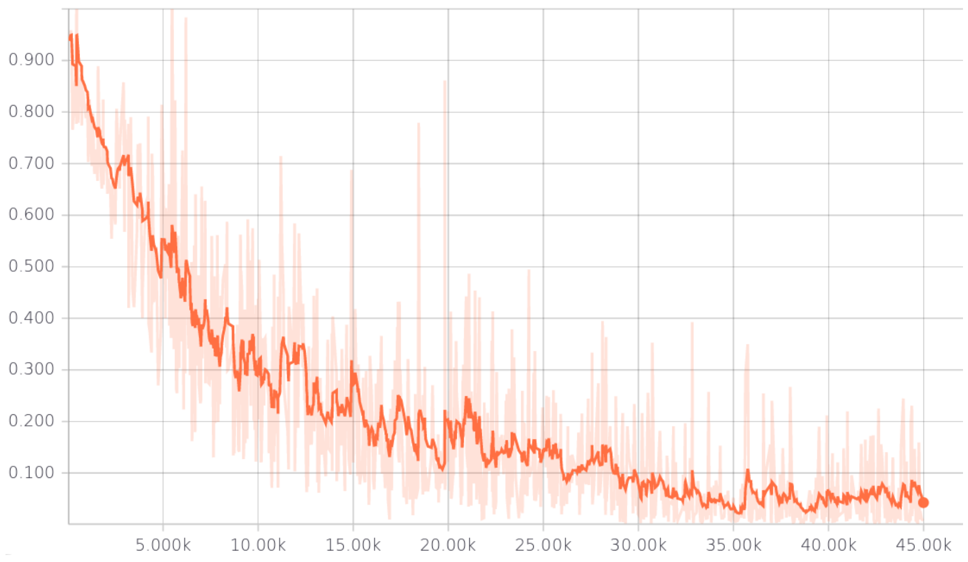


Figure 5.2: The batch hard triplet loss [40] decreasing over time during training of LuNet Lightest. The y-axis is the loss value and the x-axis is the training step. Light shade of orange is the actual score, while the dark shade of orange is the smoothened score that is easier to interpret.

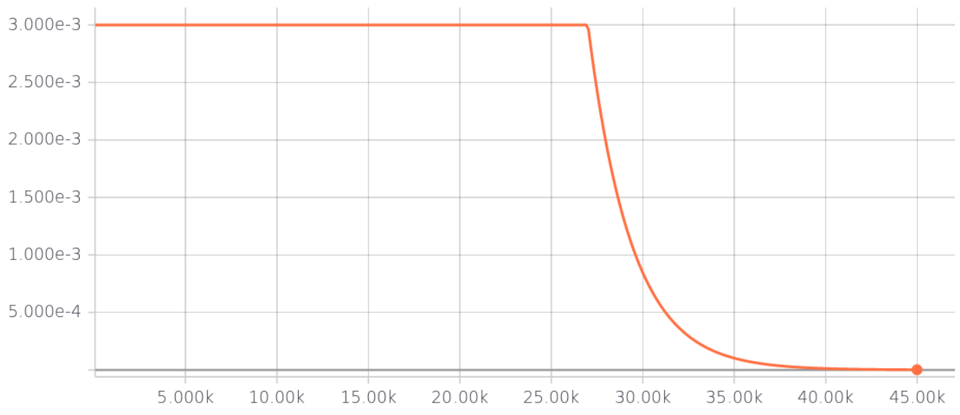


Figure 5.3: The learning rate starts decreasing after 60% of the total number of training iterations. The y-axis is the learning rate value and the x-axis is training step.

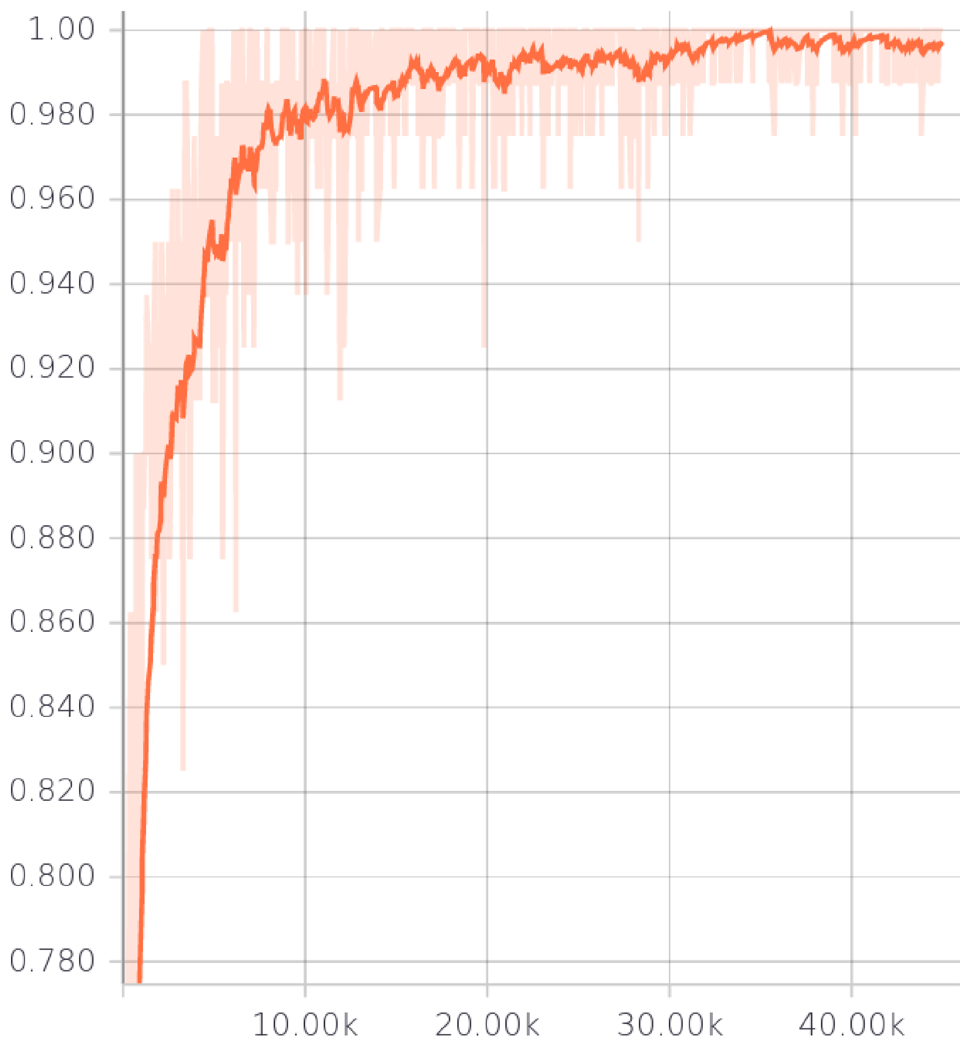


Figure 5.4: The CMC rank-1 score increasing during training. The y-axis is the CMC rank-1 score and the x-axis is the training step. Light shade of orange is the actual score, while the dark shade of orange is the smoothened score that is easier to interpret.

The evolution of the Euclidean distances between all feature vectors in a batch, as depicted in Figure 5.5, also provide information that can be useful during training. It can be observed that the maximum Euclidean distance between feature vectors is around five in the first batch and increases to a maximum value of approximately 30 towards the end of the training period. The increase in distance suggests that the network learns an embedding space where clusters of different pIDs are pushed increasingly further apart from each other. This is a positive trend because it indicates that the network can more easily distinguish between different people. During the longer training run (s_{max} = 100 000) experiments showed that the distances stopped increasing and instead stayed constant when $s > 45\,000$. The same trend can be seen in the loss function (Figure 5.2), which were the reasons why the training period was limited to 45 000 iterations.

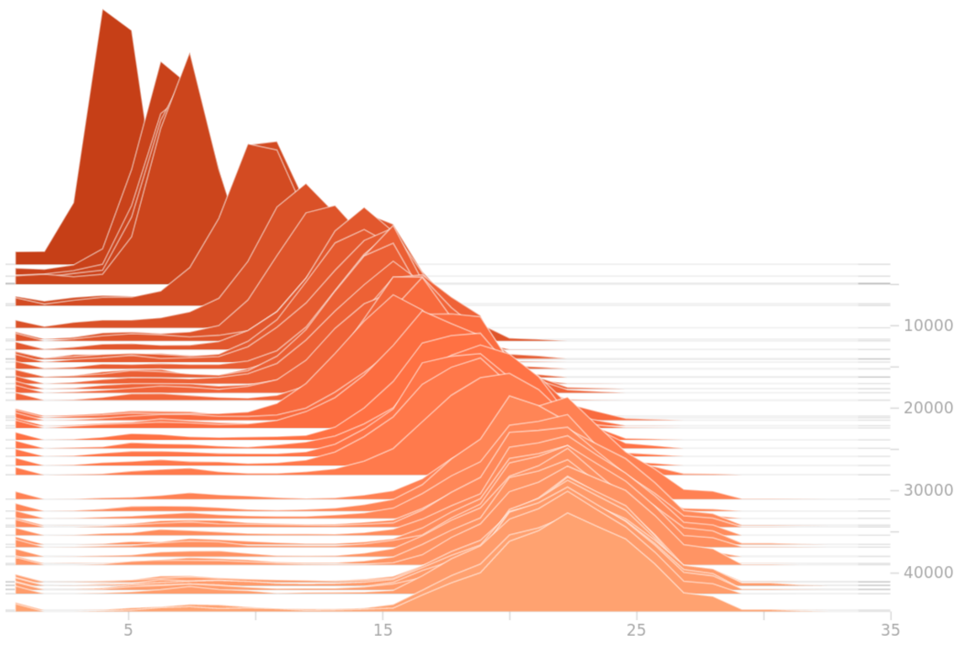


Figure 5.5: The distribution of the Euclidean distances between all feature vectors in a batch depicted in a three-dimensional figure. Each depth (y-axis, horizontal line) is one training step, the x-axis is the Euclidean distance and the height and width of each peak is the distance distribution. The upper left corner show the short distances early in the training phase, and the lower right corner shows the longer Euclidean distances towards the end of the training.

	LuNet (reproduced)	LuNet Light	LuNet Lightest
Train time (h)	11.04	6.92	10.03

Table 5.2: Number of hours needed to reach the 45 000th training iteration for each architecture.

Table 5.2 shows the time required to train each network when $s_{max} = 45\,000$. The training time for LuNet Lightest is one hour less than for the reproduced LuNet. LuNet Light can, somewhat surprisingly, complete the training stage four hours faster than the reproduced LuNet and three hours faster than LuNet Lightest. Although the exact reason for the noticeably lower training time was not investigated more closely, it is apparent that the reduced amount of both convolutional and pooling layers can have a significant impact on training times.

5.1.3 Visualization

The process of training an ANN can be very abstract. Visualizing certain network components can supplement variable monitoring by giving a visual representation of what the network learns.

Visualizing the Activation Maps

Visualizing the activation maps can provide useful insight into what the individual filters in each layer learn. It additionally provides insight into which characteristics of the input data are important when the network make its decision.

The plots of 18 randomly selected feature maps from the first, second, and final residual block in LuNet Lightest are depicted in Figures 5.6, 5.7, and 5.8, respectively. The feature maps are extracted during a forward pass in the network after the training process is completed. In CNNs in general, the early convolutional layers learn low level features, such as colors and edges, and the layers deeper in the network gradually learn higher level features that are useful for the specific problem.

This also applies for the convolutional layers in LuNet Lightest. In the first residual block, 5.6, the filters seem to have learned to differentiate between dark and light colors. In the next residual block, 5.7, the features are slightly more advanced. The filter that produced feature #14, for instance, looks to have learned to distinguish between upper-body and lower-body, which apparently (and intuitively) is a feature that is useful for the re-ID system. Feature #16, on the other hand, seems to detect the entire body. Feature #4 and feature #18 seem to represent the contour of the body. The low activation map dimension of 4×2 pixels in the final residual block (Figure 5.8) makes it impossible to extract any useful visual information. It is however likely that each of these feature maps represent a certain low-level feature that is useful to differentiate between people. Figures 8.1 and 8.2 in the appendix provide visualization of the feature maps in all residual blocks in LuNet Light.

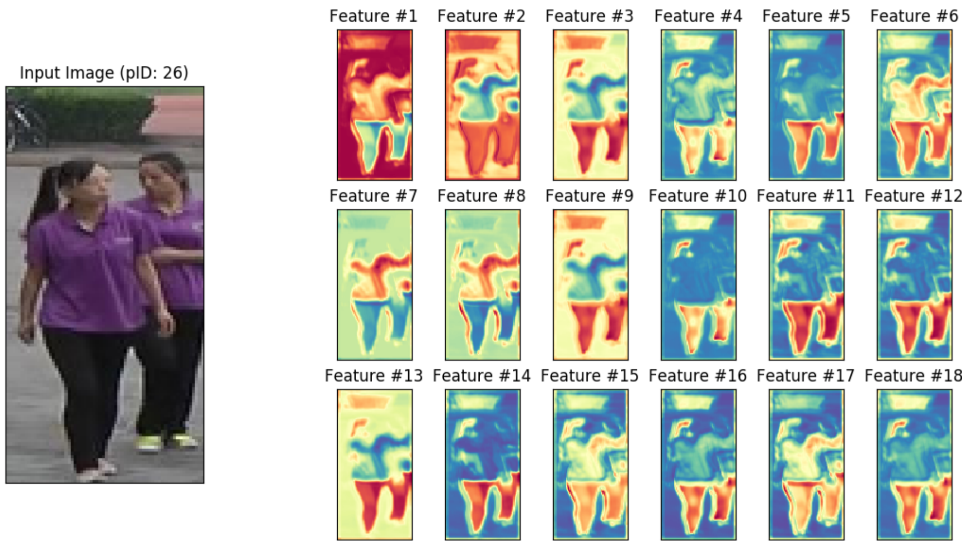


Figure 5.6: Visualization of 18 randomly selected activation maps produced by the 3×3 convolution in the first residual block (res #1) in LuNet Lightest.

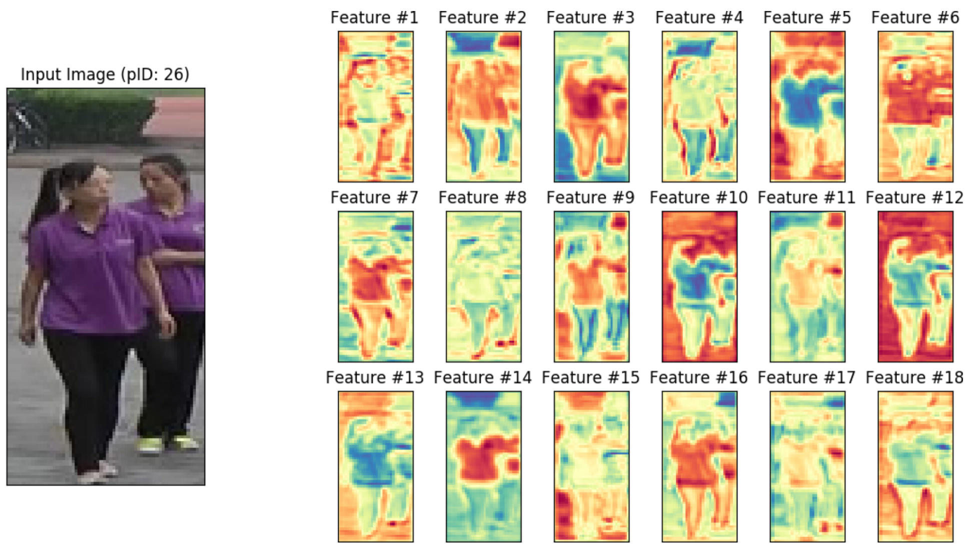


Figure 5.7: Visualization of 18 randomly selected activation maps produced by the 3×3 convolution in the second residual block (res #4) in LuNet Lightest.

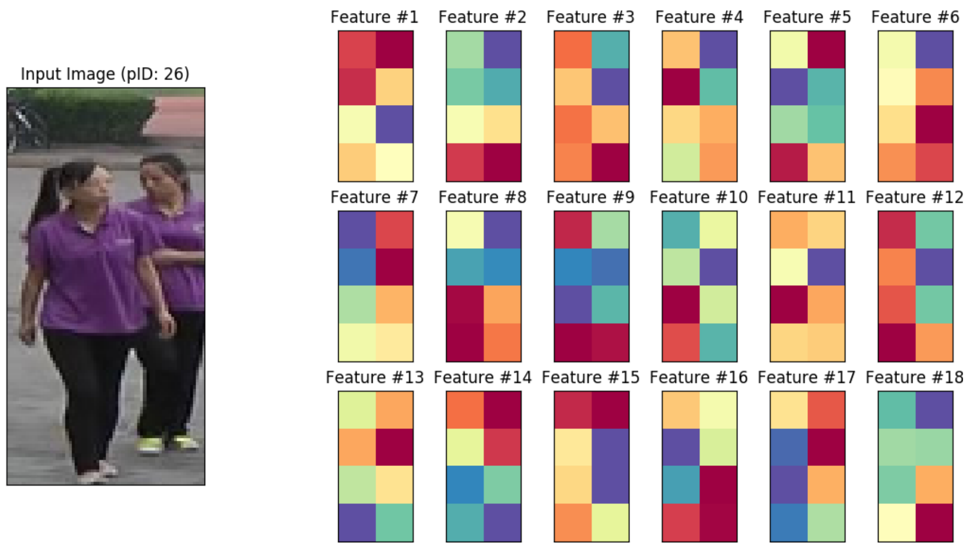


Figure 5.8: Visualization of 18 randomly selected activation maps produced by the 3×3 convolution in the seventh residual block (res #12) in LuNet Lightest.

Visualizing the Feature Embedding Space

It can be useful to plot the feature vectors to get a visual idea of how good the network is able to separate between the different pIDs. The network should group feature vectors representing the same pID close together while separating feature vectors that represent different pIDs.

Directly plotting the feature vectors in a 128-dimensional space would give little visual value. Instead, the t-SNE [76] technique was used to map the embeddings to a three-dimensional space. t-SNE is a non-linear and iterative algorithm that reduces the feature dimensionality.

Figure 5.9 shows the three-dimensional t-SNE plot of the 636 gallery persons in the MARS dataset [135] represented by their pID. Each data point is a feature vector representing one video of a person, and data points with the same color and pID are videos of the same person, whereas data points with dissimilar colors and pID are videos of different persons. It can be observed that the network consistently groups feature vectors belonging to the same person close together (e.g. the dark blue cluster of pID 272 on the right hand side). In addition to clustering feature vectors from the same persons together, there is a good amount of inter-class distance between most clusters, which indicates that the network has learned to separate videos of different persons. There are, however, some pIDs that are more difficult for the network to tell apart, such as the dark green cluster of pID 466 in the top right corner that seem to contain at least one pink feature vector depicting pID 468. Note that this figure was created in the three-dimensional space, meaning that the two-dimensional depiction is somewhat difficult to interpret accurately.

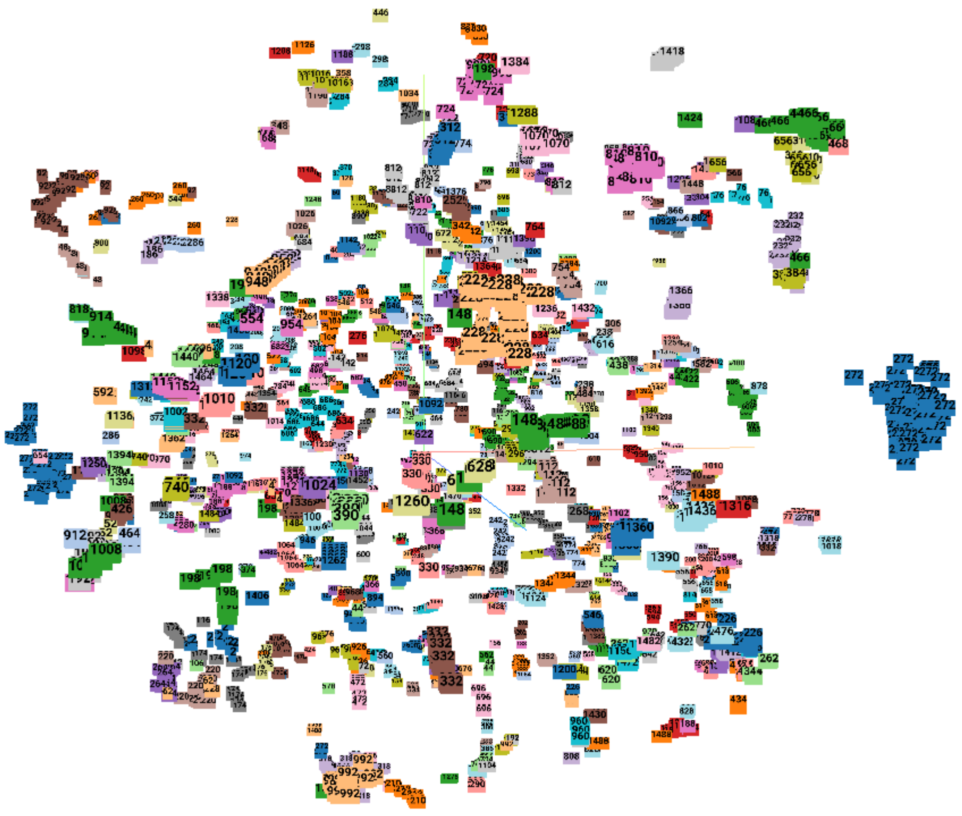


Figure 5.9: Visualization of the 128-dimensional feature embeddings using t-SNE [76]. Each data point is a video sequence of a gallery person in the MARS dataset.

5.2 Evaluation Using the MARS Evaluation Protocol

All versions of LuNet are evaluated in accordance to the MARS evaluation protocol [135]. Although this evaluation protocol is arguably not the most suitable for robotic applications, it allows us to fairly compare the performance of the three different versions of LuNet (Section 5.2.2). This evaluation protocol is additionally used to compare the results obtained by the proposed re-ID models to the results of the state-of-the-art re-ID systems (Section 5.2.3).

5.2.1 The MARS Evaluation Protocol

In the MARS evaluation protocol [135], the dataset is split into three parts: 1) the training set, 2) the gallery set, and 3) the probe set. The training data, which consists of 8 298 videos of 625 different people, is used solely for training. The gallery set, which is also referred to as the database, consists of 12 180 videos of 636 different people. It is important to note that there is no data overlap between the training set and the gallery set, which means that no person identities in the gallery have been used to train the network. The probe set contains 1 952 videos of 620 different people, an average of 3.1 videos per person. All identities in the probe set are also in the gallery set, but the videos in the probe set are unique (i.e. in terms of pIDs, the gallery and probe sets fully overlaps, but they contain different videos of each pID).

The MARS evaluation protocol uses two measures to evaluate performance: cumulative matching characteristics (CMC) and mean average precision (mAP).

Cumulative Matching Characteristics

Recall that given a set of probe videos, a re-ID system returns a list of the gallery identities that are believed to match the probe identities. This list is sorted by similarity, meaning that for each probe video, the pID at the first index of the output list is the closest match, while the pID at the end of the output list resembles the probe person the least.

CMC measures how frequently the correct match is found amongst the top n results, which is commonly noted as the *top-n* or *rank-n* score. A rank-1 score of 100% means that the re-ID systems assigned the correct identity to all probe persons. A rank-5 score of 100% means that the correct identity was always within the five best results. The rank-N score, N being the gallery size, will always be 100%.

CMC can be interpreted either as a single score or as a graph. The single score is measured by looking at one specific rank. In re-ID, it is common to evaluate rank-1 and rank-5 score, though reporting rank-10, rank-20 and rank-50 is also occasionally seen. The plotted CMC score, known as the *CMC curve*, is the plot of each rank, and is also commonly found in re-ID research.

Mean Average Precision

Since the gallery contains multiple videos of each person, the CMC score does not fully describe the system performance. For instance, a true match may be found at the top of the rank list, but there may be another true match at rank 40. mAP is the average of precision values at the ranks where there is a correct match, averaged across the number of probes. It is defined as:

$$mAP = \frac{\sum_{p=1}^{P} AP(p)}{P} \tag{5.2}$$

where P is the total number of probes and $AP(p)$ is the average precision at rank p. The closer all the true matches are to rank-1, the higher the mAP score. A mAP score of 100% means that all g gallery instances that match the probe pID are always found amongst the top g scores. Unlike the CMC curve, mAP only provides one single score and cannot be seen as a graph.

5.2.2 Comparing LuNet (reproduced), LuNet Light, and LuNet Lightest

Table 5.3 lists the results obtained by the reproduced LuNet, LuNet Light, and LuNet Lightest using the MARS evaluation protocol. The three scores in each column correspond to [*mAP, rank-1 CMC, rank-5 CMC*]. Experiments are conducted with single-image features, mean features, and max-pool features (Section 4.3.2) for both the probe and gallery, resulting in $3 \cdot 3 = 9$ different configurations. It can be observed that using the mean features on both the gallery and probe outperforms the other combinations by a significant margin in all three network architectures, which corresponds to the results obtained by Hermans et al. [40].

The scores of the reproduced LuNet, LuNet Light, and LuNet Lightest are consistently highly similar both in terms of mAP, rank-1, and rank-5, with a deviation of only a few percentage points at most. Despite having the least convolutional layers, LuNet Lightest obtains slightly higher scores than LuNet Light on all configurations of feature combinations. The reproduced LuNet and LuNet Lightest are highly similar, deviating with less than one percent most of the time. Since the combination of using the mean feature on both the query and gallery yielded the best scores, all remaining experiments are conducted with the mean features unless otherwise is noted.

Figure 5.10 shows the CMC curve of the three network architectures in one plot. In accordance to the scores in Table 5.3, it can be observed that LuNet (reproduced) and LuNet Lightest obtain very similar results, while LuNet Light scores slightly lower.

Pool method probe→gallery	LuNet (re-produced)			LuNet Light			LuNet Lightest		
single→single	51.3	69.9	86.0	47.7	67.9	83.9	50.5	70.7	85.8
single→max	38.7	62.1	76.4	34.7	56.6	73.8	37.5	59.8	77.4
single→mean	61.7	78.4	87.8	58.8	77.1	87.1	60.9	78.8	88.5
max→single	47.6	65.5	80.9	43.7	61.5	79.5	46.2	65.3	80.7
max→max	56.4	77.5	88.7	54.4	76.5	88.6	56.1	76.7	88.8
max→mean	57.8	73.6	84.8	54.6	71.3	83.4	56.7	73.8	85.0
mean→single	56.8	76.5	89.8	53.7	74.1	89.5	56.2	77.1	89.7
mean→max	42.1	66.9	82.4	38.9	63.2	80.6	41.4	65.0	82.1
mean→mean	**67.8**	**84.9**	**93.8**	**65.8**	**83.7**	**88.7**	**67.2**	**84.9**	**92.9**

Table 5.3: Evaluation of the three CNNs using different feature combination methods. "single" is the single-image feature, "mean" is the mean feature, and "max" is the max-pooled feature. The three numbers in each row correspond to [*mAP(%), rank-1 CMC(%), rank-5 CMC(%)*]. Higher numbers are better. Best and worst result for each CNN are marked in **bold** and underlined, respectively. Cyan colored rows highlights experiments where the pooling method is the same in the probe images and the gallery images. Test time augmentation was used in all experiments.

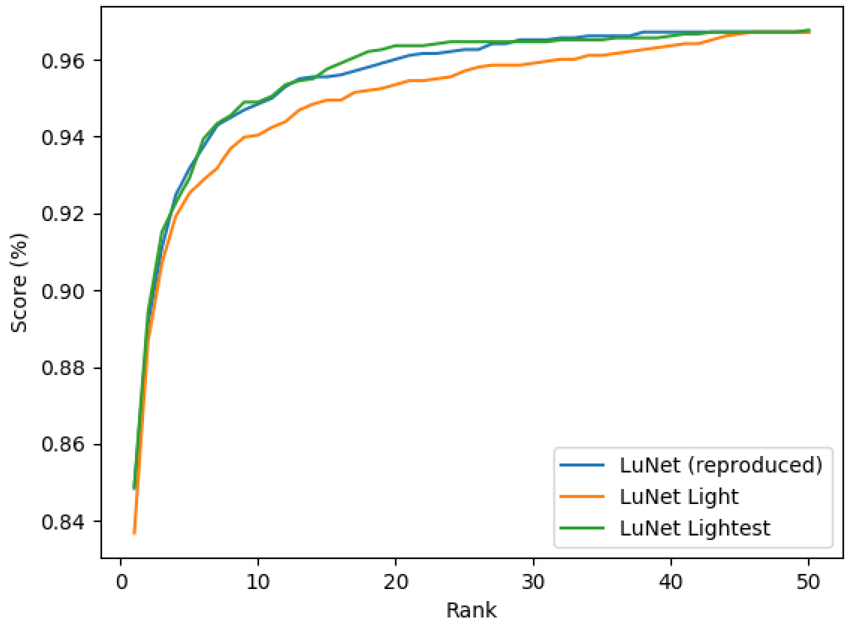


Figure 5.10: Top 50 CMC scores (rank-1 to rank-50) of the reproduced LuNet, LuNet Light, and LuNet Lightest using the mean features both on the probe and gallery data.

Test-Time Data Augmentation

Test-time data augmentation was used to produce the results in Table 5.3 and Figure 5.10. Before being fed to the network, the center crop and four corner crops of all images, along with the horizontally flipped versions are extracted (see Figure 5.11). This converts every input image into ten different images. After these have been processed by the network, the average of all ten feature vectors is used. Since the network was presented with similar crops during training (Section 4.2.4), intuitively using the average of ten crops should give better results than simply using the original image.

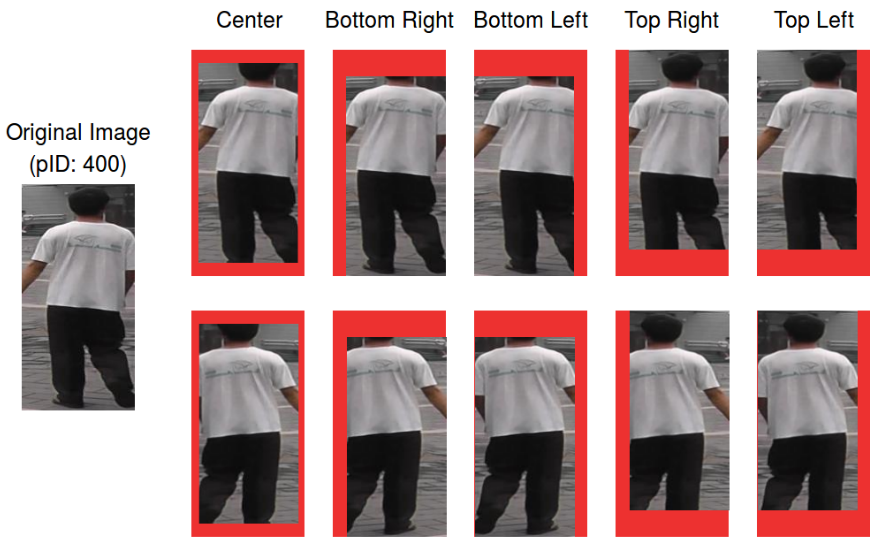


Figure 5.11: Illustration of the test-time crop and flip augmentation. The top row shows the five different crops used and the bottom row shows their horizontally flipped equivalents. The red background illustrates the area that has been removed in each crop.

The downside with test-time data augmentation, however, is that it requires a tenfold increase in the number of forward passes through the network. This will severely reduce the overall system efficiency. In this thesis it is therefore argued that test-time data augmentation should not be applied for a robotic re-ID system.

To investigate the impact of test-time augmentation on the performance of the re-ID model, the same experiments are reproduced without using the ten crops. Instead, only the center crop is passed through the network. Table 5.4 summarizes the scores obtained using only the center along with the relative difference between experiments with and without test-time augmentation. The results show that the scores have only decrease by a small margin, mostly by less than one percent. This small decrease in performance is arguably neglectable when considering how much it impacts efficiency. Remaining experiments where test-time data augmentation was used are indicated by "aug_{tt}".

	mAP (%) ↑	rank-1 (%) ↑	rank-5 (%) ↑
LuNet (reproduced)	67.1 (-0.7)	83.7 (-1.2)	92.8 (-1.0)
LuNet Light	65.3 (-0.5)	83.3 (-0.4)	92.6 (+0.1)
LuNet Lightest	66.3 (-0.9)	84.2 (-0.7)	93.0 (-0.1)

Table 5.4: Evaluation of the three networks without test-time data augmentation. Numbers in parentheses is the difference between the relative difference in score compared to the runs where test-time data augmentation was used (scores from table 5.3). Mean features was used in all experiments. "↑" indicates greater number is better.

5.2.3 Comparing With the State-Of-The-Art

Using the MARS evaluation protocol, the reproduced LuNet, LuNet Light, and LuNet Lightest are compared to the top-performing re-ID models and the scores are summarized in Table 5.5. Except for the benchmark score [135], only the re-ID systems achieving equal or better scores than any of the three LuNets are considered. Multiple observations can be made from these results. First, all versions of LuNet perform significantly better than the original implementation by Hermans et al. [40]. This is likely due to the different approach in building the batches (see Section 5.1.1), which appears to have significantly improved the training process. It may also be caused by differences in the amount of training iterations, which was set to $s_{max} = 25\,000$ by Hermans et al. [40], whereas all experiments in this thesis use $s_{max} = 45\,000$. This suggests that the network can continue to learn better feature representations after the 25 000th training iteration.

Second, the scores of both the reproduced LuNet, LuNet Light, and LuNet Lightest are competitive with the leading state-of-the-art results. Despite this, the architectures of all LuNets, especially LuNet Lightest, are significantly shallower and more lightweight than the top-performing re-ID systems developed by Zheng et al. [135], Almazán et al. [3], and Wu et al. [120]. These results suggest that it is not always necessary to use very deep architectures in order to obtain state-of-the-art performance. It is also a good indication that re-ID models possibly can be designed lightweight enough to suit the needs of mobile robots.

	mAP (%) ↑	rank-1 (%) ↑	rank-5 (%) ↑
Baseline [135]	49.3	68.3	82.6
Almazán et al. [3]	**79.7**	**85.8**	**96.5**
Wu et al. [120]	67.4	80.8	92.1
TriNet [40]	67.7	79.8	94.1
LuNet [40]	60.5	75.6	89.7
LuNet (reproduced) (aug_{tt})	67.8	84.9	93.8
LuNet Light (aug_{tt})	65.3	83.3	92.6
LuNet Lightest (aug_{tt})	67.2	84.9	92.2

Table 5.5: State-of-the-art scores on the MARS dataset evaluation protocol. *Baseline* is the results Zheng et al. [135] obtained along with the release of the MARS dataset. "↑" indicates greater number is better. Best result is marked in bold, results obtained in this thesis are marked by cyan background.

5.3 Evaluation in an Open-World Setting

5.3.1 Common Restrictive Assumptions in Re-ID

Most re-ID evaluation protocols, including the MARS evaluation protocol, make several assumptions that are highly unrealistic for a robotic re-ID system. First, it is assumed that every probe person is guaranteed to be present in the gallery. This assumption does not hold for robots that are expected to operate in dynamic,real-world environments where there may be multiple people that the robot has not seen before. Removing this assumption increases the problem difficulty because, in addition to performing regular re-ID (matching pIDs), the system has to separate strangers from acquaintances and learn on-the-fly. Second, re-ID is always performed with a gallery of fixed size. This assumption is also unrealistic for robotic applications because robots will need to automatically register people it interacts with. A thorough description of common assumptions in re-ID may be found in Background Chapter 2.3.

5.3.2 Open-World Evaluation Metrics

Open-world re-ID, as discussed in the background (Section 2.3.6), investigates the re-ID problem in a setting where the probe person's identities are not guaranteed to be present in the gallery. In an open-world re-ID problem, the CMC and mAP metrics that are normally used in re-ID are not capable of adequately evaluating performance because they do not take the unknown identities into account. Therefore, Zheng et al. [140] introduced the following set of evaluation metrics specifically designed for the open-world setting:

$$True\ Target\ Recognition(TTR) = \frac{\#TTQ}{\#TQ} \tag{5.3}$$

$$False\ Target\ Recognition(FTR) = \frac{\#FNTQ}{\#NTQ} \tag{5.4}$$

where the authors define *target* people as people amongst the probes that are also present in the gallery and *non-target* people as people amongst the probes that are not present in the gallery. Finally, TQ, NTQ, TTQ and $FNTQ$ are defined as:

$$\begin{aligned}
TQ &= \{probe\ target\ images\ from\ target\ people\},\\
NTQ &= \{probe\ non\text{-}target\ images\ from\ non\text{-}target\ people\},\\
TTQ &= \{probe\ target\ images\ that\ are\ verified\ as\ one\\
&\quad of\ the\ target\ people\},\\
FNTQ &= \{probe\ non\text{-}target\ images\ that\ are\ verified\ as\ one\\
&\quad of\ the\ target\ people\}.
\end{aligned}$$

Although TTR and FTR are adequate measures for the open-world reID problem, these metrics' relevancy for a mobile robotic re-ID application are very limited for two reasons. First, these open-world evaluation metrics are applied in a setting where the number of probes, which consists of both targets and non-targets, is much larger than the gallery, which consists of targets only [68, 140, 144]. This is a common scenario in surveillance, where there are typically only a few gallery persons of interest (targets), but the potentially large network of surveillance cameras can result in a large collection of probe people (both targets and non-targets). In robotics, however, it is likely that the gallery, which contains information about all individuals the robot has had an interaction with, is large, whereas the number of probes, which is the collection of all people currently present in the robot's environment, is likely to only contain a few people. For example, a companion robot that operates in people's homes, will have a gallery consisting of the person(s) living in that home, and the gallery will grow slowly as relatives and friends visit. It is arguably likely that the robot will meet only one or very few strangers (new people visiting) at a time, which is the opposite scenario compared to what the open-world evaluation metrics are designed for [140]. Mobile robots that operate outdoors or amongst larger crowds, such as in shopping malls [45], will certainly meet more strangers compared to companion robots, but the amount is arguably not comparable to surveillance systems that may have weeks or months worth of surveillance video captured by multiple cameras.

Second, the open-world evaluation metrics are reported as the True Target (Equation 5.3) Rate in percent against the False Target Rate (Equation 5.4). In other words, these metrics represent how accurately targets are verified when a varying amount of non-targets are allowed amongst the retrieved persons (i.e. how many non-targets are acceptable for each target retrieved). This is useful in a surveillance application where such a system could guarantee to find target(s) along with a pool of n non-targets. A person could then manually examine the pool of n person identities instead of the full collection of probes. On a mobile robot that needs to perform re-identification in real-time, relying on a human agent is certainly not a feasible option. Instead, to better depict a realistic situation for a mobile robots, the evaluation metrics need to represent how accurately the robot can distinguish between acquaintances and strangers, and how precisely the acquaintances can be correctly identified. To that end, this thesis proposes a new set of evaluation metrics specifically designed to evaluate re-ID systems intended for robotic applications.

5.3.3 Approaching Evaluation From a Robotic Perspective

This thesis presents a set of evaluation metrics for a re-ID system in a robotic setting. The proposed evaluation protocol is based on evaluation metrics commonly used for binary classifiers.

Defining Binary Evaluation Metrics for Robotic Re-ID

In a binary classification problem, the true value of each data point can be either a *positive instance* or a *negative instance*, and the classifier's objective is to assign the correct value, either positive or negative, to each data point. As depicted in Figure 5.12, this yields four different possibilities; *true positive* (TP, positive instance correctly classified as positive), *false positive* (FP, negative instance incorrectly classified as positive), *false negative* (FN, positive instance incorrectly classified as negative), and *true negative* (TN, negative instance correctly classified as negative). The sum of TPs and FNs is always equal to the total amount of positive data points, whereas the sum of FPs and TNs is always equal to the total amount of negative data points. In a perfect binary classifier, all outputs are either TPs or TNs.

A robotic re-ID system classifies each probe person as either a) an acquaintance (positive) or b) a stranger (negative). If a person is an acquaintance, the system additionally has to determine the person's identity, which can either be:

- a^+: the reID system correctly determined the person's identity (TP), or
- a^-: the re-ID system incorrectly determined the person's identity (FP).

Note that only considering a is equivalent to the regular re-ID problem that has been have examined thus far; whereas, adding b is the additional objective of an open-world robotic re-ID system.

Using these possible outcomes, the binary conditions for the proposed evaluation protocol are defined as follows:

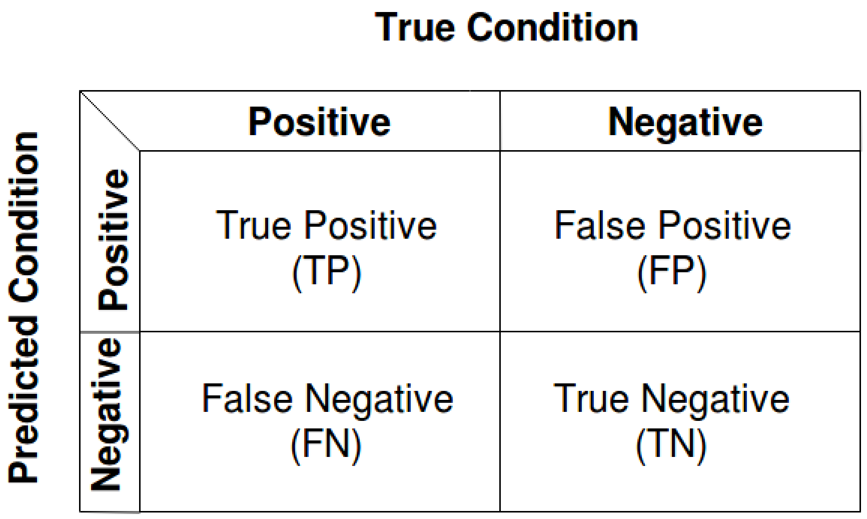


Figure 5.12: Evaluation metrics for binary classifiers are commonly derived based on four data attributes: True Positives (TP), False Positives (FP), False Negatives (FN), and True Negatives (TN).

$$\begin{aligned}
TP &= \{acquaintance\ classified\ as\ an\ acquaintance\ with\ correct\ pID\}, \\
TN &= \{stranger\ classified\ as\ a\ stranger\}, \\
FP &= \{stranger\ classified\ as\ an\ acquaintance\}, \\
FN &= \{acquaintance\ classified\ as\ a\ stranger\ or\ as\ an\ acquaintance\ but \\
&\quad\ with\ incorrect\ pID\}.
\end{aligned}$$

Threshold Values

A threshold value, t, is used to control whether the re-ID system classifies a probe person as an acquaintance or a stranger. When the probe person's feature vector is compared to all gallery feature vectors, the probe person is classified as a stranger if the distance between the probe feature vector and the best matching gallery feature vector is larger than t. If this distance is less or equal to t, however, the pID of the best matching gallery person is assigned to the probe person. In other words, for each probe person:

$$p = \begin{cases} stranger & if\ sim(p, g_n) > t \\ acquaintance & if\ sim(p, g_n) \leq t \end{cases} \tag{5.5}$$

where p is one probe person, g_n is the gallery person that has been assigned the shortest distance (highest similarity) to p by the re-ID model, and $sim(p, g_n)$ is the distance as returned by the re-ID system. If p is classified as an acquaintance, the additional step of deciding the correct pID is performed. This procedure is done according to:

$$pID_p = pID_{g_n} \tag{5.6}$$

where pID_{g_n} is the person ID of the gallery person with the highest similarity to the probe person.

Experiment are conducted with threshold values in the range $t \in [0,\ 15)$, and a step of 0.125, resulting in 120 different values. The selection of this range is simply based on the fact that the Euclidean distance between any two feature vectors is always positive, and the observation during experiments that the re-ID system never assigns a distance greater than 15 to any two feature vectors.

Gallery Size

In addition to experimenting with various threshold values, experiments are conducted with a varying amount of unique identities present in the gallery. Varying the gallery size (G_{size}) affects the ratio of acquaintances vs. strangers in the probe set, which simulates different conditions and scenarios a mobile robot may operate in. In the default setting, G_{size} = 100% (636 unique pIDs), all probe persons are present in the gallery and therefore all probes are acquaintances. The smaller G_{size}, the fewer probes are present in the gallery, resulting in more probe persons being strangers instead of acquaintances.

To be more scenario-specific, experiments conducted with a large gallery, $G_{size} \geq 70\%$, depict the typical scenario for household robots that operate in relatively closed environments in patients homes, and that therefore do not meet strangers regularly. Experiments conducted with smaller gallery sizes, $G_{size} \leq 30\%$, on the other end, realistically represent the environment of robots that meet strangers very frequently, such as shopping mall robots [45]. Experiments using gallery sizes in-between, $30\% < G_{size} < 70\%$, are realistic for robots that are regularly surrounded by both strangers and acquaintances. An example is TritonBot [113] that operated in the entrance hall in a building at UC San Diego and regularly met the same professors and alumni that enter and leave the building on a daily basis, but also frequently interacted with one-time visitors.

In all experiments, the gallery size ranges between $G_{size} \in [10\%,\ 90\%]$ with a step of 10%, resulting in nine different gallery sizes. The size of the probe set is kept constant to 620 unique pIDs, as defined by the MARS train/test data split [135].

Balanced Accuracy

Balanced accuracy is calculated based on the number of TPs and TNs, and is defined as follows:

$$B_{ACC} = \frac{(\frac{TP}{P} + \frac{TN}{N})}{2} \tag{5.7}$$

.
The balanced accuracy measure is the percentage of the data that has been correctly classified as either TPs or TNs. It is similar to accuracy, but better suited for datasets where the data is unbalanced. It is a well-suited measure to evaluate a robotic re-ID system because varying the gallery size results in different ratio between strangers and acquaintances, i.e. unbalanced data.

Receiver Operating Characteristic Curve

The receiver operating characteristic (ROC) curve is a plot that illustrates the true positive rate (TPR) along the y-axis against the false positive rate (FPR) along the x-axis as the threshold is varied. TPR and FPR are defined as follows:

$$TPR = \frac{TP}{P} = \frac{TP}{TP + FN} \tag{5.8}$$

$$FPR = \frac{FP}{N} = \frac{FP}{FP + TN} \tag{5.9}$$

The TPR and FPR measures are similar to TP and FP, but instead of providing absolute values, TPR and FPR show the percentage of the detected TPs and FPs. Similarly with TP and FP, a good threshold value should maximize TPR while simultaneously minimize FPR. Figure 5.13 depicts three different ROC curves, depicting a) the perfect ROC curve, b) a god ROC curve and c) a poor ROC curve as the threshold varies. Systems with ROC curves similar to Figure 5.13b are capable of achieving high TPR while maintaining low FPR. Note that unlike balanced accuracy that considers both TPs and TNs, the ROC is only concerned with how well positives (acquaintances) are classified. Although not used in the proposed evaluation metrics, the true negative rate $TNR = \frac{TN}{N}$ and the false negative rate $FNR = \frac{FN}{P}$ may provide useful statistics and are therefore included as plots in the appendix Section 8.4 along with plots of the TPR and the FPR.

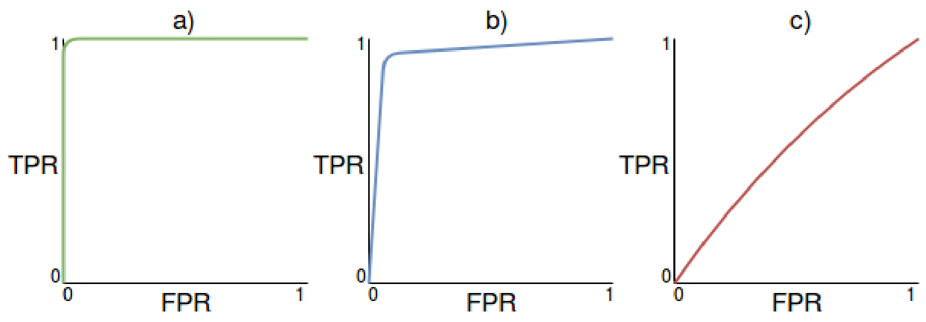


Figure 5.13: Example of a) a perfect ROC curve, b) a system with a good ROC curve, and c) a system with a bad ROC curve as a threshold varies. Systems with good ROC curves can achieve high TPR and low FPR simultaneously.

Results

As previously seen, LuNet Lightest performs comparably to the reproduced LuNet in terms of CMC and mAP (see Sections 5.2.2 and 5.2.3), and contains considerably fewer learnable parameters. As a result, LuNet Lightest is most suitable for robotic applications. Therefore, LuNet Lightest was selected to be evaluated using the newly proposed evaluation metrics.

Figures 5.14, 5.15, 5.16, and 5.17 respectively show the distributions of TPs, TNs, FPs, and FNs when varying G_{size} and t according to the afore discussed value ranges. Recall that a robust system should maximize the amount of TPs and TNs and simultaneously minimize the amount of FPs and FNs. In this evaluation, the number of correctly classified probes is the main objective, and discussing the TPs (Figure 5.14) and TNs (Figure 5.15) is therefore the main focus.

The graphs of the binary conditions show that varying the threshold value t greatly affect the performance of the re-ID system. When t is close to zero (left hand side in all figures), the similarity score between any two feature vectors is greater than t, thus all probes will be classified as strangers (negative) (see Equation 5.5). All TNs are therefore detected, but none of the acquaintances, TPs, have been detected. When t is close to its maximum value, 15, all similarity scores will be smaller than t, meaning that all acquaintances (TPs) get detected while the system no longer detects strangers (TNs). For threshold values that are less extreme, say $t = 6$, for instance, both strangers and acquaintances will be detected, but there will also be a certain amount of both FPs and FNs.

Secondly, let us discuss how the gallery size G_{size} affects the distribution of TPs and TNs. Since a smaller gallery is equivalent to having less acquaintances and more strangers in the probe set, the maximum amount of TPs is low for a small gallery and greater for a large gallery (see right hand side of Figure 5.14). Similarly, the maximum amount of TNs is greater for a small gallery and lower for a large gallery. Furthermore, note from the y-axis in Figure 5.14 and 5.15 that the maximum amount of TNs is around 1 750, while the maximum amount of TPs is slightly lower, around 1 500. This is due to the fact that the system easily can detect all strangers simply by classifying all probe persons as strangers (which can be achieved by using low threshold values, see Equation 5.5), but correctly classifying all acquaintances, however, additionally requires finding the correct pID. In other words, to reach 1 750 TPs, the re-ID system would need a rank-1 CMC score of 100%.

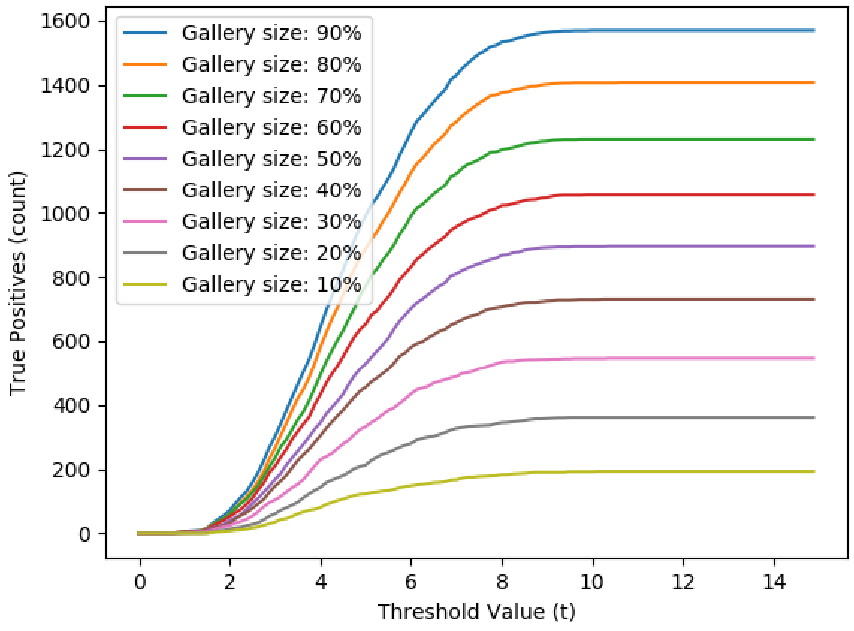


Figure 5.14: Total number of TPs (y-axis) for various threshold values (x-axis) and varying gallery sizes.

The results presented this far show that t greatly affects how many TPs and TNs the robotic re-ID system is capable of classifying. To find the optimal threshold values at varying gallery sizes, the balanced accuracy measure (B_{ACC}) was used.

Figure 5.18 shows the balanced accuracy measure for varying gallery size and threshold values. The peak of each curve corresponds to the threshold value that yields the highest number of TPs and TNs. Table 5.6 summarizes the statistics at the peak points of each curve (the same information is provided as a plot in Figure 8.3 in the appendix). The balanced accuracy score is higher when the gallery is small. This is because a smaller gallery is synonymous with more strangers, and correctly detecting strangers (TNs) is easier than detecting acquaintances and additionally assigning the correct pID (TPs). However, while the gallery size certainly impacts the balance accuracy measure, the lowest observed score is B_{ACC} = 82.2%

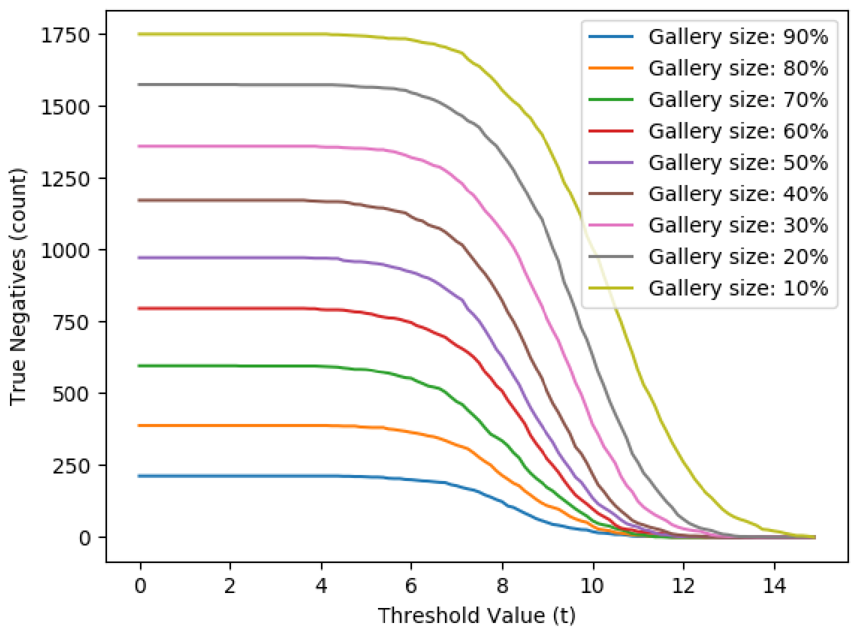


Figure 5.15: Total number of TNs (y-axis) for various threshold values (x-axis) and varying gallery sizes.

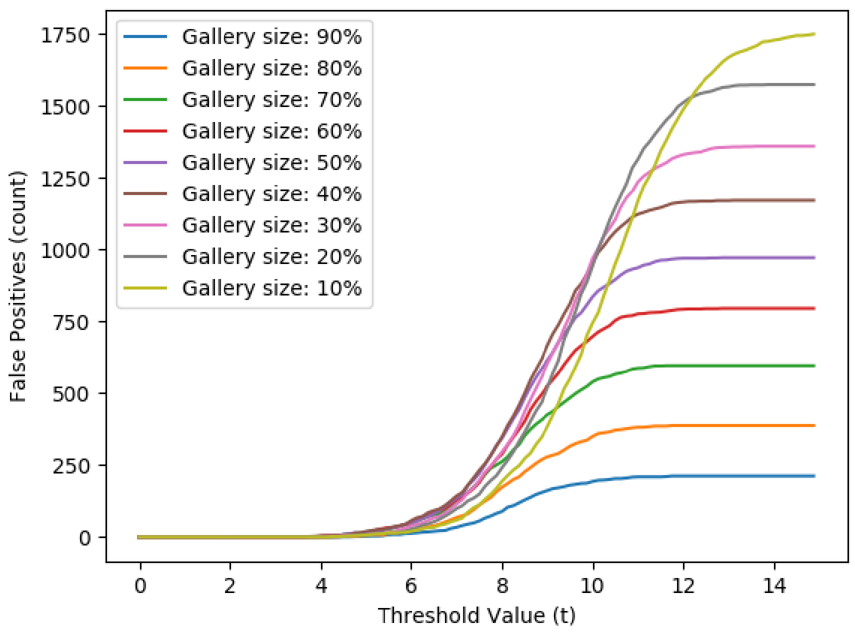


Figure 5.16: Total number of FPs (y-axis) for various threshold values (x-axis) and varying gallery sizes.

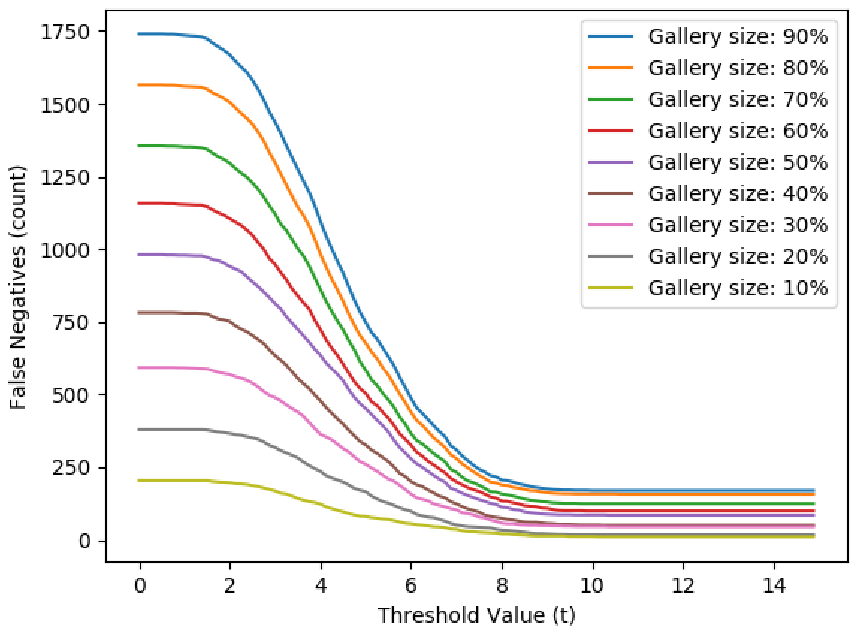


Figure 5.17: Total number of FNs (y-axis) for various threshold values (x-axis) and varying gallery sizes.

when G_{size} = 10%, suggesting that the proposed re-ID system is good at distinguishing strangers from acquaintances and simultaneously assigning the correct pID to acquaintances.

G_{size}	10%	20%	30%	40%	50%	60%	70%	80%	90%
$B_{ACC} \uparrow$	**92.4**	88.4	87.4	86.8	85.4	85.4	83.7	84.3	$\underline{82.2}$
t	7.5	7.0	7.125	6.875	7.0	6.75	6.25	6.625	6.25

Table 5.6: The optimal threshold value t that maximizes the balanced accuracy score B_{ACC}(%) at varying gallery size G_{size}. "↑" indicates greater number is better. The highest score is marked in **bold** and the lowest score is underlined.

Table 5.6 suggests that $6.250 \leq t \leq 7.125$ are the optimal threshold values for the different gallery sizes. This threshold should, however, be tuned based on the typical environment of the robot. For instance, for a household robot that is likely to mostly interact with acquaintances it is arguably more important to correctly classify acquaintances than to detect strangers, and re-ID systems on such robots could therefore benefit from having a should have a higher threshold (see Equation 5.5). Robots operating in more crowded areas, on the other hand, could probably benefit from a re-ID system with a lower threshold to ensure that strangers are less frequently mislabeled as acquaintances. In other words, the threshold should be used to adjust the tradeoff between TPs and TNs based on the robot's typical environment.

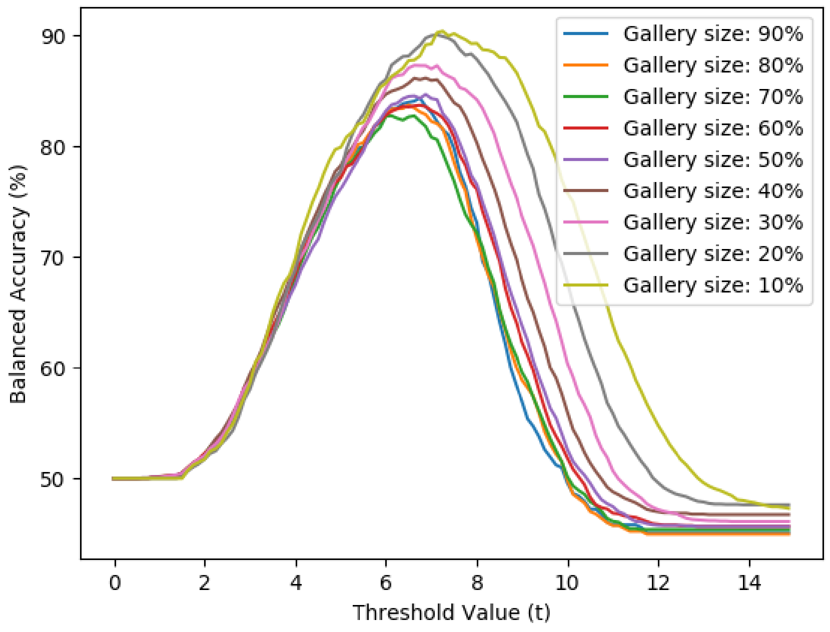


Figure 5.18: Balanced accuracy curves at varying gallery sizes and threshold values.

In addition to analyzing the balanced accuracy, the ROC curve, as seen in Figure 5.19, is used to investigate the robotic re-ID performance with the same varying values for t and G_{size}. It can be observed that, for all gallery sizes, the ROC curves look similar to the left hand side of Figure 5.13, indicating that the re-ID system is capable of reaching high TPR while maintaining low FPR. The curves corresponding to $G_{size} = 10\%$ and $G_{size} = 20\%$ (and to some extent $G_{size} = 20\%$ and $G_{size} = 30\%$) have somewhat better characteristics which is evident by their curvation that start at a higher TPR. This supports the results obtained with the balanced accuracy measure, validating that the re-ID system can more easily detect strangers than correctly classify acquaintances.

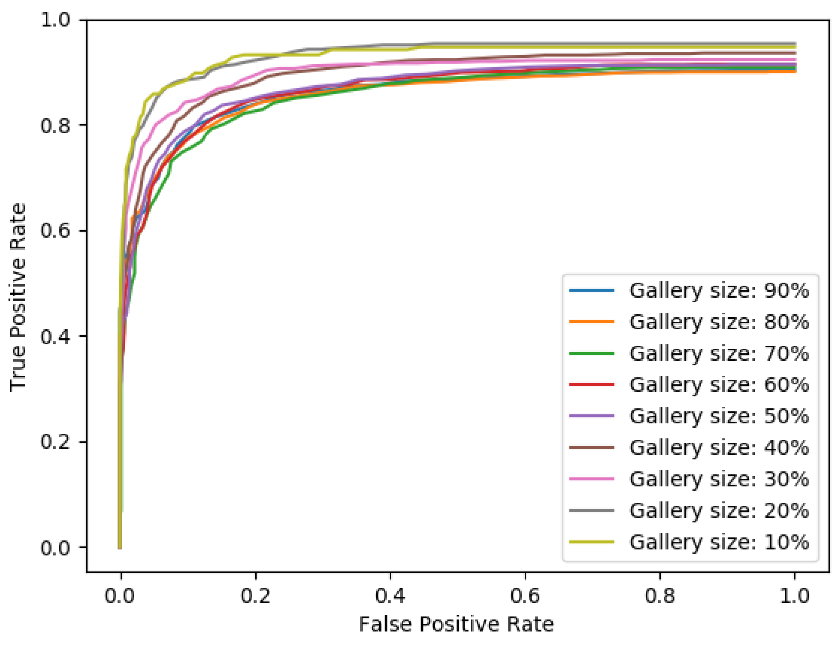


Figure 5.19: The ROC curve at varying gallery size and threshold values.

5.3.4 System Efficiency

The results discussed this far suggest that the re-ID model performs well both on the MARS benchmark and on the newly proposed evaluation metrics in a robotic setting. These metrics do, however, only evaluate system performance, and are of little importance if the system is unable to operate efficiently in terms of frame rates. High processing speed is crucial for mobile robots to operate smoothly in accordance to the users' expectations. Systems that do not reach real-time requirement may have to be processed remotely on more powerful hardware, which restricts the robots to operating in areas with network connection, and can also, in the worst of cases, lead to increased privacy risks. As discussed in Section 2.3.9, CV re-ID systems rarely report efficiency, resulting in little knowledge whether or not re-ID systems can operate on robots that require real-time feedback.

In light of this, the efficiency in terms of frame rate is evaluated for both the reproduced LuNet, LuNet Light, and LuNet Lightest. First, the frame rate at which each network could process data is measured. These measurements were obtained by feeding videos through each network frame by frame and measuring the average amount of video frames the networks could process in the course of a second. Next, the frame rate for the entire re-ID pipeline consisting of the feature extraction and the feature matching is measured. These measurements were obtained by using the entire MARS gallery set of 636 unique pIDs, and combining the frame rate of the CNN with the frame rate of the feature matching.

	LuNet (reproduced)	LuNet Light	LuNet Lightest
GPU ↑	75.8 (67.4)	**109.7 (96.3)**	78.8 (71.6)
CPU ↑	32.8 (30.7)	32.9 (30.8)	**36.4 (33.9)**
Single core CPU ↑	13.4 (12.7)	11.9 (11.3)	**16.8 (15.7)**

Table 5.7: Time efficiency of each network when using GPU, only CPU, and only one out of eight CPU cores. Numbers in parentheses are the efficiency of the entire re-ID system (i.e. network processing and feature matching). All numbers are reported in FPS. "↑" indicates greater number is better. Best result marked in **bold**.

Table 5.7 shows the processing time of all three re-ID architectures measured in FPS. In each column, the leftmost number is the frame rate of the CNN and the rightmost number inside parentheses is the efficiency of the entire re-ID pipeline. For each network architecture, the frame rates with the use of a GPU, with the use of CPU only, and finally with the use of only one out of the eight CPU cores are evaluated (see the introduction Section 1.3 and appendix Section 8.1 for details regarding hardware and software specifications). Since mobile robots typically have hardware constraints due to cost, lack of memory, and limited power resources, it is important to evaluate the frame rate on a CPU and single core CPU to simulate these conditions.

The frame rate is often considered to be real-time if it exceeds the frame rate of the camera sensor, which is typically 30 FPS. When using the GPU, all three LuNets exceed this requirement by a large margin. Similarly to the training times (Table 5.2 in Section 5.1.2), LuNet Light is 44.7% more efficient than LuNet (reproduced) and 40.8% more efficient than LuNet Lightest despite its higher amount of learnable parameters (see Table 4.2 in Section 4.2.1). This is because it consists of less consecutive operations (fewer layers) but more operations in each layer (larger input volume), which are very efficiently calculated by the GPU.

When using CPU and single core CPU, however, LuNet Lightest is notably more efficient than LuNet (reproduced) and LuNet Light. This is due to LuNet Lightest containing fewer residual blocks and learnable parameters. All architectures achieve real-time frame rates with a CPU, but the efficiency is significantly lower on a single core CPU. However, 15.7 FPS as obtained by LuNet Lightest on a single core CPU is still relatively high on considering the hardware limitations. With high camera frame rates, such as 30 FPS, two consecutive image frames are likely to be very similar, meaning that it could be possible to only perform the re-identification on a subset of all video frames to retain real-time performance.

In addition to evaluating frame rate, Table 5.8 provides an overview

Network architecture	Memory requirement pr image (MB) ↓
LuNet (reproduced)	33.9
LuNet Light	36.4
LuNet Lightest	**29.7**

Table 5.8: Memory requirement for one forward pass in the network. "↓" indicates lower number is better. Best result is marked in **bold**.

over the amount of memory required by each network to process one video frame. The table shows that LuNet Lightest is the most lightweight in terms of memory requirements, requiring 12.4% less memory than the reproduced LuNet. These results show that a re-ID model with near state-of-the-art benchmark performance can achieve high frame rates at low memory requirements, and therefore be suitable for real-time robotic applications.

5.4 Discussion

This chapter has presented various experiments conducted with the reproduced LuNet, LuNet Light, and LuNet Lightest. First, the comparison of the three different LuNets and the results obtained by the inventors of the original LuNet [40] show that it is possible to ease the network training by slightly modifying the batch selection, which allows for enhanced performance even with shallower network architectures. It is also shown that all three versions of LuNet achieve near state-of-the-art performance in terms of mAP and CMC on the MARS evaluation protocol, despite being significantly more lightweight than other top-performing systems [3, 120].

Second, this thesis points out why the existing evaluation protocol for an open-world re-ID setting cannot adequately evaluate a robotic re-ID system. A new set of evaluation measures to more realistically assess robotic re-ID systems are therefore proposed. Experiments using these measures show that the ratio between strangers and acquaintances affects the re-ID performance. Nevertheless, when using the optimal threshold value, LuNet Light achieves up to 92.4% balanced accuracy on the newly proposed evaluation metrics. This indicates that the proposed system is capable of tackling the re-ID problem also in an open-world, robotic environment.

Finally, the memory requirement and system efficiency is evaluated for all three LuNets. The results show that all three architectures reach beyond real-time frame rates both with and without the use of a GPU. With a single core CPU, which may be representative for more lightweight and hardware-constrained robots, LuNet Lightest processes 15.7 FPS, which is significantly more than both the reproduced LuNet and LuNet Light. These efficiency measurements emphasize the importance of designing lightweight system architectures to ensure they can perform adequately on mobile robots.

Chapter 6

Conclusion

This research aimed at creating a person re-identification system for mobile robots. A lightweight re-ID system has been proposed and compared to state-of-the-art research. This work shows that it is possible to achieve near state-of-the-art performance on the MARS dataset benchmark without using deep CNNs that require large amounts of computational resources. Furthermore, it has been shown that small modifications in the batch selection for the batch hard triplet loss can ease network training and improve re-ID performance on the MARS dataset.

Multiple limitations and assumptions found in CV re-ID and on re-ID systems for mobile robots have been pointed out in this thesis. These include strictly relying on facial cues, needing preliminary information about individuals to re-identify (either by requiring an enrollment phase, only being able to re-identify a small number of specific individuals, or assuming that all people are acquaintances), or not meeting requirements of real-time frame-rates and low memory usage. Evaluation on the MARS benchmark assumes that all persons to re-identify are acquaintances, and does therefore not adequately describe how well the re-ID system performs on a mobile robot.

Therefore, in addition to evaluating the proposed system on the MARS dataset benchmark, this thesis presented a set of evaluation measures that are appropriate to assess how well the re-ID system performs when removing these restrictions. First, evaluations were conducted with a varying number of strangers and acquaintances. The results show that the re-ID system largely is capable of distinguishing between strangers and acquaintances, in addition to associating the correct identification number to the acquaintances. This demonstrates that the proposed system can adequately tackle typical unconstrained, open-world environments for a mobile robot.

Second, additional experiments were conducted to assess the memory requirements and processing speed of the re-ID system. Measurements show that the network architecture is sufficiently lightweight to achieve real-time frame rate requirements, even without the use of a GPU.

To summarize, this thesis demonstrates that it is possible to design lightweight person re-identification systems that meet the requirements of mobile robots. The proposed model is a framework that can be integrated on mobile robots, thereby providing an enabler for more polite and personalized robot behavior, and in addition enlargening and enhancing robots' repertoire of social skills. It is hoped that the findings of this thesis may contribute into developing more lightweight and time efficient person re-ID models to better fit the needs of mobile robotics.

Chapter 7

Future Work

Two possible directions for future research, namely deployment on a mobile robot (Section 7.1) and suggestions for potential model improvements (Section 7.2), are identified and discussed in this chapter.

7.1 Deployment on a Mobile Robot

Although the conducted experiments suggest that the proposed re-ID system is suitable for mobile robots, deploying and testing the model on a mobile robot was out of the scope of this project. A natural continuation of this work would therefore be to integrate the re-ID model with an automatic person detector and person tracker, and deploy the full system on a mobile robot. Doing so would allow observing how the system works in real-world conditions. Furthermore, data could be collected and used to evaluate the model in accordance with the proposed robotic re-ID measures. A comparison of results obtained using data recorded from a mobile robot and results obtained using the MARS dataset [135] would provide valuable insight into how well the model truly performs in real conditions. Moreover, combining the re-ID model with an automatic person detector and tracker adds more computations to the overall system, meaning that the frame rate is likely to be lower than reported in this work. Therefore, experiments involving combinations of various detectors and trackers, and possibly running the re-ID model on only every nth frame in order to achieve desired frame rate, could be viable paths for further investigation.

Deploying the re-ID system on a mobile robot may require tuning or adjustments to the matching threshold. The optimal threshold may vary based on the robot's environment, and could possibly diverge over time if the robot operates in a range of various environments. To accommodate for these changes, a threshold value that dynamically adjusts itself based on the distances between clusters in the embedding space may be more appropriate than a naive, static threshold value.

7.2 Possible Model Extensions and Modifications

To fairly compare the model to the state-of-the-art on the MARS benchmark, only the training set of the MARS dataset [135] was used to train the model in all experiments. However, two possible approaches to optimize the model for a real application instead of benchmark evaluations are identified. First, training on the entire MARS dataset [135] would double the amount of training data and arguably improve performance. It is also possible to combine multiple datasets to form an even larger amount of training data. Training on multiple datasets require relatively small modifications to the existing system, but may have a significant impact on the model's performance on real applications.

Second, many mobile robots have depth video readily available in addition to color video. Depth data can, for instance, be used to improve the frame rate in robot perception systems [13]. However, due to the limited amount of available depth data in re-ID datasets, an unexplored area of research is how to optimally combine depth and RGB data in a re-ID pipeline. Experimenting with depth data recorded from a mobile robot could provide valuable insight about whether depth data can improve re-ID performance.

Another possible improvement to the model could be using a more sophisticated way of combining features. For instance, instead of averaging the feature vectors of all images in a video, an RNN can be added after the CNN to extract temporal features. However, combining an RNN with the Siamese architecture could result in higher processing times due to the added layers, hyperparameters, and learnable parameters. These variables therefore need to be taken into consideration to preserve real-time frame-rate.

Chapter 8

Appendix

8.1 Hardware and Software

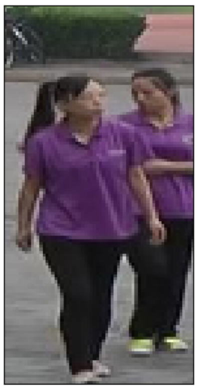
- Laptop: Dell Inspiron 15 7000 series (2016)
- CPU: Intel Core i7-6700HQ, 2.60GHz, 8 cores, 64 bit
- GPU: Intel GeForce GTX 960M, 33MHz, 64 bit
- OS: Ubuntu 16.04.5 LTS
- Python 3.5.2
- Tensorflow-GPU 1.11.0
- Tensorflow-tensorboard 0.4.0
- Tensorflow Embedding Projector
- GNU Emacs 24.5.1
- Matplotlib 3.0.0

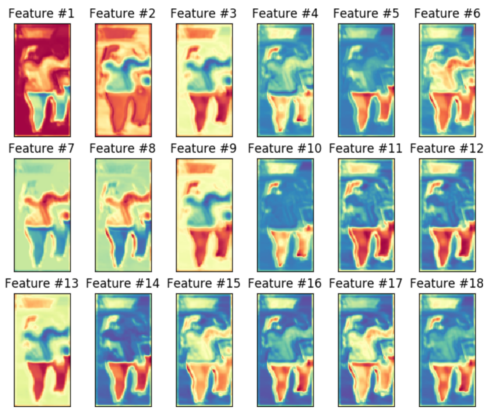
8.2 Detailed Network Architectures

Layer type	LuNet [40]	LuNet Light	LuNet Lightest
conv #1	128×7×7×3	128×7×7×3	128×7×7×3
res #1	128,32,128	128,32,128	128,32,128
pool #1	pool 3×3, stride 2, pad 1	pool 3×3, stride 2, pad 1	pool 3×3, stride 2, pad 1
res #2	128,32,128	128,32,128	-
res #3	128,32,128	128,32,128	-
res #4	128,64,256	128,64,256	128,64,256
pool #2	pool 3×3, stride 2, pad 1	pool 3×3, stride 2, pad 1	pool 3×3, stride 2, pad 1
res #5	256,64,256	-	-
res #6	256,64,256	-	256,64,256
pool #3	pool 3×3, stride 2, pad 1	-	pool 3×3, stride 2, pad 1
res #7	256,64,256	256,64,256	-
res #8	256,64,256	256,64,256	256,64,256
res #9	256,128,512	256,128,512	256,128,512
pool #4	pool 3×3, stride 2, pad 1	pool 3×3, stride 2, pad 1	pool 3×3, stride 2, pad 1
res #10	512,128,512	512,128,512	512,128,512
res #11	512,128,512	512,128,512	-
pool #5	pool 3×3, stride 2, pad 1	pool 3×3, stride 2, pad 1	pool 3×3, stride 2, pad 1
res #12	512×(3×3×512), 128×(3×3×512)	512×(3×3×512), 128×(3×3×512)	512×(3×3×512), 128×(3×3×512)
FC #1	1024×512	1024×512	1024×512
batch-norm #1	512	512	512
LReLU	512	512	512
FC #2	512×128	512×128	512×128

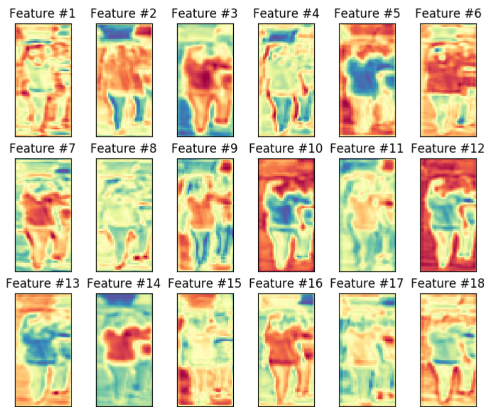
Table 8.1: The architecture of the three networks considered in this project. "conv" is a single convolutional layer (#in channels, filter height, filter width, #out channels). "res" is a bottleneck residual block. For each number (n_a, n_b, n_c) in the "res" rows, the operation consists of one 1×1 convolution from n_a to n_b channels, followed by a 3×3 convolution, and finally a 1×1 convolution from n_b to n_c channels. "res12" deviates from the standard bottleneck block, and the filter sizes are given in the table. "pool" are max-pooling layers. "FC" are fully connected layers ($n_a \times n_b$), which reduce the embedding dimensionality from n_a to n_b features. "batch-norm" are batch normalization layers and "LReLU" is the leaky rectified linear unit activation function [75].

8.3 Visualization of Activation Maps

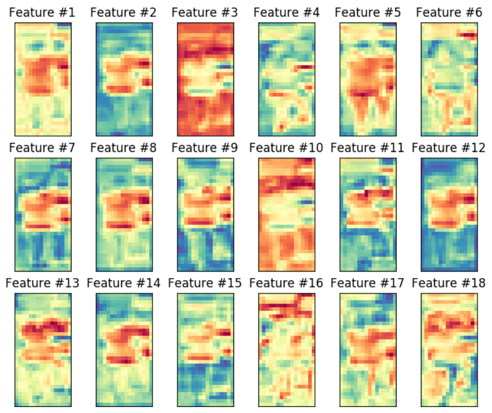

a) Input image (pID: 26)



b) res #1 (128 $\times$ 64)



c) res #4 (64 $\times$ 32)



d) res #6 (32 $\times$ 16)

Figure 8.1: One input image (a) and 18 corresponding activation maps from b) the first residual block, c) the second residual block, and d) the third residual block. The dimensions of the respective activation maps are on the format $H \times W$. Continued in Figure 8.2.

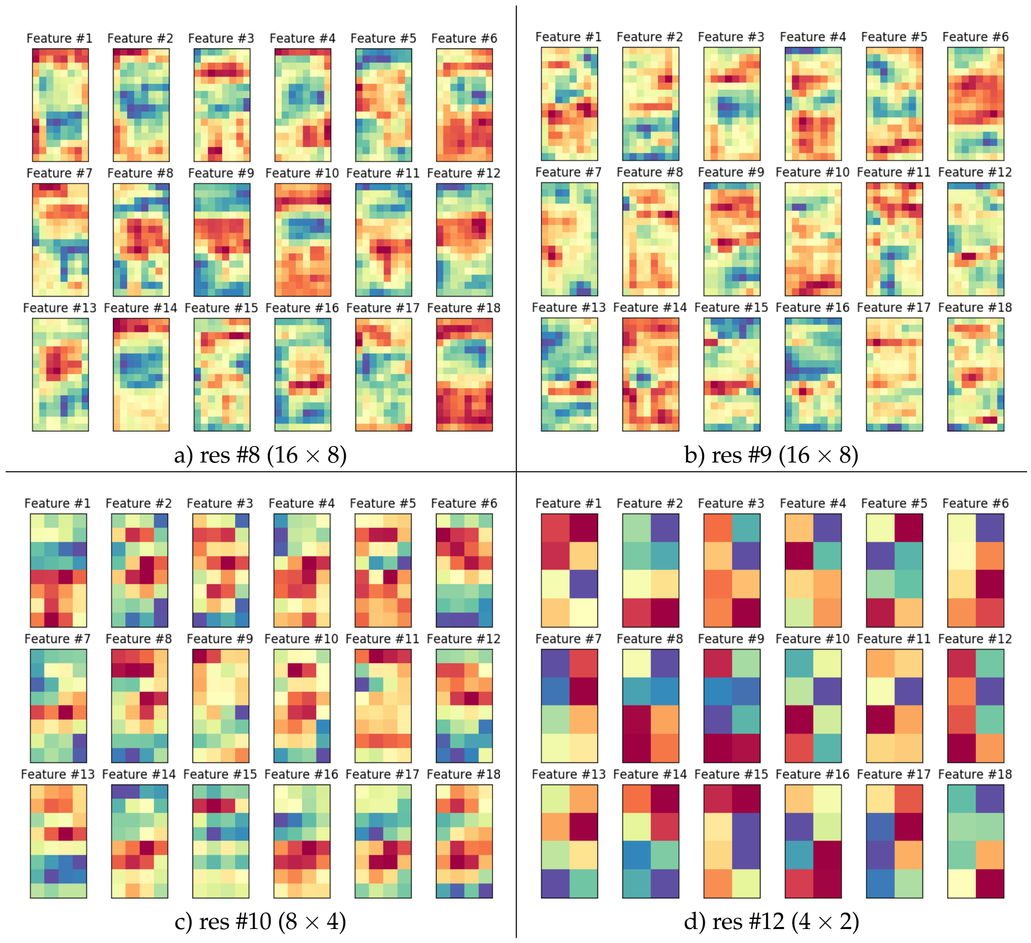


Figure 8.2: Continuation of Figure 8.1. 18 activation maps from a) the fourth residual block, b) the fifth residual block, c) the sixth residual block, and d) the seventh and final residual block. The dimensions of the respective activation maps are on the format $H \times W$.

8.4 Additional Results

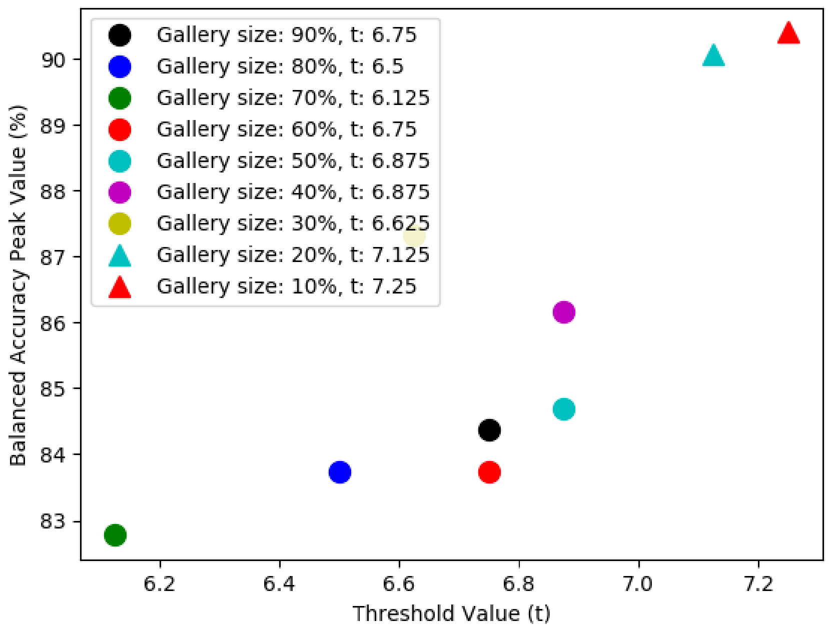


Figure 8.3: Peak balanced accuracy at varying gallery sizes and threshold values.

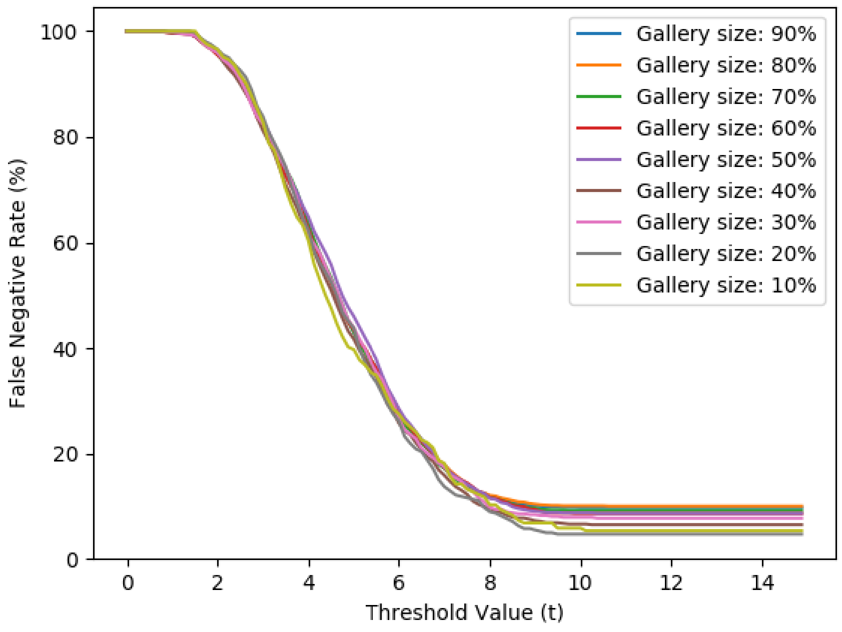


Figure 8.4: The FNR at varying gallery size and threshold values.

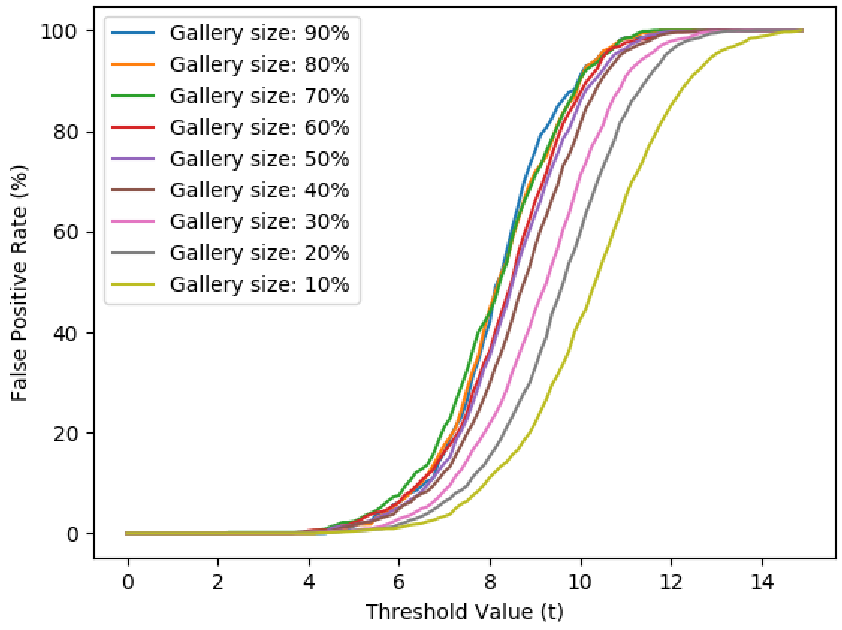


Figure 8.5: The FPR at varying gallery size and threshold values.

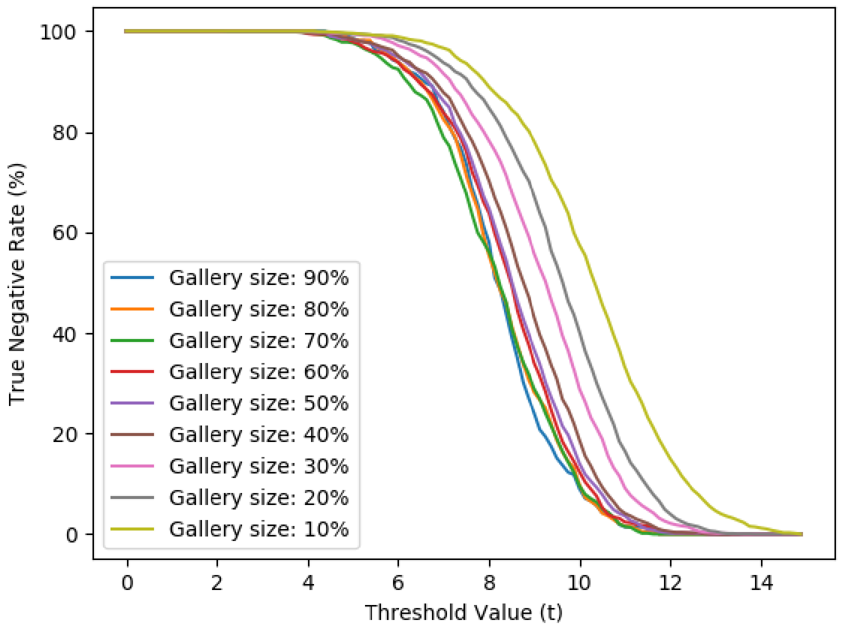


Figure 8.6: The TNR at varying gallery size and threshold values.

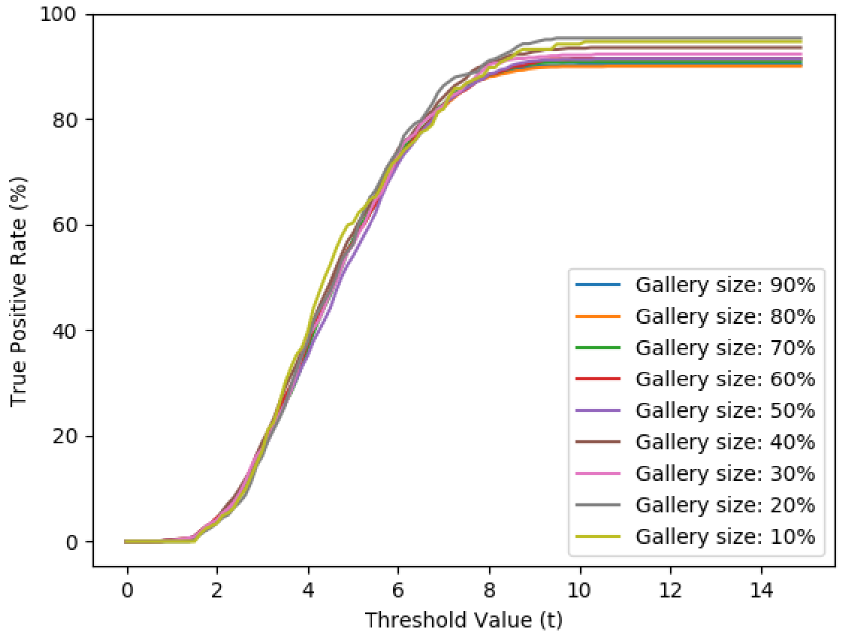


Figure 8.7: The TPR at varying gallery size and threshold values.

Bibliography

[1] The Center for Advanced Spatial Technologies (CAST). *Geospatial Modeling & Visualization*. [Online; accessed February 17, 2019]. 2019. URL: `https://developer.microsoft.com/en-us/windows/kinect`.

[2] Ejaz Ahmed, Michael Jones, and Tim K Marks. "An improved deep learning architecture for person re-identification." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 3908–3916.

[3] Jon Almazan et al. "Re-ID done right: towards good practices for person re-identification." In: *arXiv preprint arXiv:1801.05339* (2018).

[4] Fernando Alonso-Martin et al. "Identification and distance estimation of users and objects by means of electronic beacons in social robotics." In: *Expert Systems with Applications* 86 (2017), pp. 247–257.

[5] Victor Alvarez-Santos et al. "Feature analysis for human recognition and discrimination: Application to a person-following behaviour in a mobile robot." In: *Robotics and Autonomous Systems* 60.8 (2012), pp. 1021–1036.

[6] Ning An et al. "Online context-based person re-identification and biometric-based action recognition for service robots." In: *2017 29th Chinese Control And Decision Conference (CCDC)*. IEEE. 2017, pp. 3369–3374.

[7] Alberto Torres Angonese and Paulo Fernando Ferreira Rosa. "Multiple people detection and identification system integrated with a dynamic simultaneous localization and mapping system for an autonomous mobile robotic platform." In: *2017 International Conference on Military Technologies (ICMT)*. IEEE. 2017, pp. 779–786.

[8] Xiang Bai et al. *Deep-Person: Learning Discriminative Deep Features for Person Re-Identification*. 2017. arXiv: `1711.10658 [cs.CV]`.

[9] Davide Baltieri, Roberto Vezzani, and Rita Cucchiara. "3dpes: 3d people dataset for surveillance and forensics." In: *Proceedings of the 2011 Joint ACM Workshop on Human Gesture and Behavior Understanding*. ACM. 2011, pp. 59–64.

[10] B. I. Barbosa et al. "Re-identification with RGB-D sensors." In: *First International Workshop on Re-Identification*. Oct. 2012.

[11] Nicola Bellotto and Huosheng Hu. "A bank of unscented kalman filters for multimodal human perception with mobile service robots." In: *International Journal of Social Robotics* 2.2 (2010), pp. 121–136.

[12] Nicola Bellotto and Huosheng Hu. "Multisensor data fusion for joint people tracking and identification with a service robot." In: *2007 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE. 2007, pp. 1494–1499.

[13] Darren M Chan, Angelique Taylor, and Laurel D Riek. "Faster Robot Perception Using Salient Depth Partitioning." In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2017, pp. 4152–4158.

[14] Bao Xin Chen, Raghavender Sahdev, and John K Tsotsos. "Integrating stereo vision with a cnn tracker for a person-following robot." In: *International Conference on Computer Vision Systems (ICCV)*. Springer. 2017, pp. 300–313.

[15] Bao Xin Chen, Raghavender Sahdev, and John K Tsotsos. "Person following robot using selected online ada-boosting with stereo camera." In: *2017 14th Conference on Computer and Robot Vision (CRV)*. IEEE. 2017, pp. 48–55.

[16] De Cheng et al. "Person re-identification by multi-channel parts-based cnn with improved triplet loss function." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 1335–1344.

[17] Dong Seon Cheng et al. "Custom pictorial structures for re-identification." In: *British Machine Vision Conference (BMVC)*. Vol. 1. 2. Citeseer. 2011, p. 6.

[18] Grzegorz Cielniak and Tom Duckett. "People recognition by mobile robots." In: *Journal of Intelligent & Fuzzy Systems* 15.1 (2004), pp. 21–27.

[19] Serhan Cosar, Claudio Coppola, Nicola Bellotto, et al. "Volume-based Human Re-identification with RGB-D Cameras." In: *Proceedings of the 12th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP) 2017*. 2017, pp. 389–397.

[20] Ju Dai et al. "Video person re-identification by temporal residual learning." In: *IEEE Transactions on Image Processing* 28.3 (2019), pp. 1366–1377.

[21] Abir Das, Anirban Chakraborty, and Amit K Roy-Chowdhury. "Consistent re-identification in a camera network." In: *European Conference on Computer Vision*. Springer. 2014, pp. 330–345.

[22] Abir Das, Anirban Chakraborty, and Amit K Roy-Chowdhury. "Consistent re-identification in a camera network." In: *European Conference on Computer Vision*. Springer. 2014, pp. 330–345.

[23] Afshin Dehghan, Shayan Modiri Assari, and Mubarak Shah. "Gmmcp tracker: Globally optimal generalized maximum multi clique problem for multiple object tracking." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 4091–4099.

[24] Piotr Dollár et al. "Fast feature pyramids for object detection." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36.8 (2014), pp. 1532–1545.

[25] Pedro Domingos. "A few useful things to know about machine learning." In: *Communications of the ACM* 55.10 (2012), pp. 78–87.

[26] Markus Eisenbach et al. "User recognition for guiding and following people with a mobile robot in a clinical environment." In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2015, pp. 3600–3607.

[27] Michela Farenzena et al. "Person re-identification by symmetry-driven accumulation of local features." In: *2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2010, pp. 2360–2367.

[28] Pedro F Felzenszwalb et al. "Object detection with discriminatively trained part-based models." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32.9 (2010), pp. 1627–1645.

[29] Miha Finžgar and Primož Podržaj. "Machine-vision-based human-oriented mobile robots: A review." In: *Strojniski Vestnik/Journal of Mechanical Engineering* 63.5 (2017).

[30] Xavier Glorot and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks." In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. 2010, pp. 249–256.

[31] Mengran Gou. *The Awesome Person Re-Identification Datasets*. [Online; accessed February 21, 2019]. 2018. URL: `https://github.com/NEU-Gou/awesome-reid-dataset`.

[32] Mengran Gou et al. "DukeMTMC4ReID: A large-scale multi-camera person re-identification dataset." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2017, pp. 10–19.

[33] Douglas Gray, Shane Brennan, and Hai Tao. "Evaluating appearance models for recognition, reacquisition, and tracking." In: *Proceedings of the IEEE International Workshop on Performance Evaluation for Tracking and Surveillance (PETS)*. Vol. 3. 5. Citeseer. 2007, pp. 1–7.

[34] Douglas Gray and Hai Tao. "Viewpoint invariant pedestrian recognition with an ensemble of localized features." In: *European Conference on Computer Vision*. Springer. 2008, pp. 262–275.

[35] H-M Gross et al. “Mobile robotic rehabilitation assistant for walking and orientation training of stroke patients: A report on work in progress.” In: *2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE. 2014, pp. 1880–1887.

[36] Horst-Michael Gross et al. “ROREAS: robot coach for walking and orientation training in clinical post-stroke rehabilitation—prototype implementation and evaluation in field trials.” In: *Autonomous Robots* 41.3 (2017), pp. 679–698.

[37] Peter A Hancock et al. “A meta-analysis of factors affecting trust in human-robot interaction.” In: *Human Factors* 53.5 (2011), pp. 517–527.

[38] Kaiming He et al. “Deep residual learning for image recognition.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778.

[39] Kaiming He et al. “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification.” In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 1026–1034.

[40] Alexander Hermans, Lucas Beyer, and Bastian Leibe. “In defense of the triplet loss for person re-identification.” In: *arXiv preprint arXiv:1703.07737* (2017).

[41] Martin Hirzer et al. “Person re-identification by descriptive and discriminative classification.” In: *Scandinavian Conference on Image Analysis*. Springer. 2011, pp. 91–102.

[42] Sergey Ioffe and Christian Szegedy. “Batch normalization: Accelerating deep network training by reducing internal covariate shift.” In: *arXiv preprint arXiv:1502.03167* (2015).

[43] Bahar Irfan et al. “Multi-modal Open-Set Person Identification in HRI.” In: *2018 HRI Workshop* (2018).

[44] Alex Kacik. *Healthcare costs increasing at unsustainable pace*. [Online; accessed April 25, 2019]. 2018. URL: `https://www.modernhealthcare.com/article/20180613/NEWS/180619961/healthcare-costs-increasing-at-unsustainable-pace`.

[45] Takayuki Kanda et al. “A communication robot in a shopping mall.” In: *IEEE Transactions on Robotics* 26.5 (2010), pp. 897–913.

[46] Takayuki Kanda et al. “Interactive robots as social partners and peer tutors for children: A field trial.” In: *Human-Computer Interaction* 19.1-2 (2004), pp. 61–84.

[47] Andrej Karpathy. *Convolutional Neural Networks for Visual Recognition*. [Online: accessed April 26, 2019]. 2019. URL: `http://cs231n.github.io/convolutional-networks/`.

[48] Parija Kavilanz. *The US can't keep up with demand for health aides, nurses and doctors*. [Online; accessed April 25, 2019]. 2018. URL: `https://money.cnn.com/2018/05/04/news/economy/health-care-workers-shortage/index.html`.

[49] Nitish Shirish Keskar et al. "On large-batch training for deep learning: Generalization gap and sharp minima." In: *arXiv preprint arXiv:1609.04836* (2016).

[50] Minkyu Kim et al. "An Architecture for Person-Following using Active Target Search." In: *arXiv preprint arXiv:1809.08793* (2018).

[51] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization." In: *arXiv preprint arXiv:1412.6980* (2014).

[52] Martin Koestinger et al. "Large scale metric learning from equivalence constraints." In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2012, pp. 2288–2295.

[53] Kenji Koide and Jun Miura. "Convolutional Channel Features-Based Person Identification for Person Following Robots." In: *International Conference on Intelligent Autonomous Systems*. Springer. 2018, pp. 186–198.

[54] Kenji Koide and Jun Miura. "Identification of a specific person using color, height, and gait features for a person following robot." In: *Robotics and Autonomous Systems* 84 (2016), pp. 76–87.

[55] Lucy Kok, Caroline Berden, and Klarita Sadiraj. "Costs and benefits of home care for the elderly versus residential care: a comparison using propensity scores." In: *The European Journal of Health Economics* 16.2 (2015), pp. 119–131.

[56] Tsuyoshi Komatsubara et al. "Can a social robot help children's understanding of science in classrooms?" In: *Proceedings of the Second International Conference on Human-Agent Interaction*. ACM. 2014, pp. 83–90.

[57] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks." In: *Advances in Neural Information Processing Systems*. 2012, pp. 1097–1105.

[58] Lars Kunze et al. "Artificial Intelligence for Long-Term Robot Autonomy: A Survey." In: *IEEE Robotics and Automation Letters* 3.4 (2018), pp. 4023–4030.

[59] Stanford Vision Lab. *ImageNet*. [Online; accessed April 9, 2019]. 2016. URL: `http://www.image-net.org/`.

[60] Beom-Jin Lee et al. "Robust Human Following by Deep Bayesian Trajectory Prediction for Home Service Robots." In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 7189–7195.

[61] Dangwei Li et al. "Learning deep context-aware features over body and latent parts for person re-identification." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 384–393.

[62] Wei Li and Xiaogang Wang. "Locally aligned feature transforms across views." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2013, pp. 3594–3601.

[63] Wei Li, Rui Zhao, and Xiaogang Wang. "Human reidentification with transferred metric learning." In: *Asian Conference on Computer Vision*. Springer. 2012, pp. 31–44.

[64] Wei Li, Xiatian Zhu, and Shaogang Gong. "Person re-identification by deep joint learning of multi-loss classification." In: *arXiv preprint arXiv:1705.04724* (2017).

[65] Wei Li et al. "Deepreid: Deep filter pairing neural network for person re-identification." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2014, pp. 152–159.

[66] Wei Li et al. "Deepreid: Deep filter pairing neural network for person re-identification." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2014, pp. 152–159.

[67] Zhen Li et al. "Learning locally-adaptive decision functions for person verification." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2013, pp. 3610–3617.

[68] Shengcai Liao et al. "Open-set person re-identification." In: *arXiv preprint arXiv:1408.0872* (2014).

[69] Ji Lin et al. "Consistent-aware deep learning for person re-identification in a camera network." In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Vol. 6. 2017.

[70] Hao Liu et al. "End-to-end comparative attention networks for person re-identification." In: *IEEE Transactions on Image Processing* 26.7 (2017), pp. 3492–3506.

[71] Hao Liu et al. "Video-based person re-identification with accumulative motion context." In: *IEEE Transactions on Circuits and Systems for Video Technology* 28.10 (2018), pp. 2788–2802.

[72] Zimo Liu, Dong Wang, and Huchuan Lu. "Stepwise Metric Promotion for Unsupervised Video Person Re-Identification." In: *The IEEE International Conference on Computer Vision (ICCV)*. Oct. 2017.

[73] David G Lowe. "Object recognition from local scale-invariant features." In: *Proceedings of the seventh IEEE International Conference on Computer Vision (ICCV)*. Vol. 2. IEEE. 1999, pp. 1150–1157.

[74] Chen Change Loy, Chunxiao Liu, and Shaogang Gong. "Person re-identification by manifold ranking." In: *2013 IEEE International Conference on Image Processing (ICIP)*. IEEE. 2013, pp. 3567–3571.

[75] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. "Rectifier nonlinearities improve neural network acoustic models." In: *Proceedings of the 30th International Conference on Machine Learning*. Vol. 30. 1. 2013, p. 3.

[76] Laurens van der Maaten and Geoffrey Hinton. "Visualizing data using t-SNE." In: *Journal of Machine Learning Research* 9.Nov (2008), pp. 2579–2605.

[77] Niki Martinel and Christian Micheloni. "Re-identify people in wide area camera network." In: *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*. IEEE. 2012, pp. 31–36.

[78] Niall McLaughlin, Jesus Martinez del Rincon, and Paul Miller. "Recurrent convolutional network for video-based person re-identification." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 1325–1334.

[79] Niall McLaughlin, Jesus Martinez del Rincon, and Paul C Miller. "Person Reidentification Using Deep Convnets With Multitask Learning." In: *IEEE Transactions on Circuits and Systems for Video Technology* 27.3 (2017), pp. 525–539.

[80] Matteo Munaro et al. "One-shot person re-identification with a consumer depth camera." In: *Person Re-Identification*. Springer, 2014, pp. 161–181.

[81] Payam Nikdel, Rakesh Shrestha, and Richard Vaughan. "The Hands-Free Push-Cart: Autonomous Following in Front by Predicting User Trajectory Around Obstacles." In: *Robotics and Automation* (2018).

[82] Deqiang Ouyang, Yonghui Zhang, and Jie Shao. "Video-based person re-identification via spatio-temporal attentional and two-stream fusion convolutional networks." In: *Pattern Recognition Letters* 117 (2019), pp. 153–160.

[83] P Pala et al. "Person re-identification from depth cameras using skeleton and 3D face data." In: *Proceedings of the 11th Eurographics Workshop on 3D Object Retrieval*. Eurographics Association. 2018, pp. 95–101.

[84] Shyamal Patel and Johanna Pingel. *Introduction to Deep Learning: What Are Convolutional Neural Networks?* [Online; accessed February 6, 2019]. 2017. URL: `https://www.mathworks.com/videos/introduction-to-deep-learning-what-are-convolutional-neural-networks--1489512765771.html`.

[85] Xuelin Qian et al. "Multi-Scale Deep Learning Architectures for Person Re-Identification." In: *The IEEE International Conference on Computer Vision (ICCV)*. Oct. 2017.

[86] Omar A Islas Ramirez et al. "Robots learning how and where to approach people." In: *2016 25th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*. IEEE. 2016, pp. 347–353.

[87] Shaoqing Ren et al. "Faster r-cnn: Towards real-time object detection with region proposal networks." In: *Advances in Neural Information Processing Systems*. 2015, pp. 91–99.

[88] International Federation of Robotics (IFR). *Service Robots*. [Online; accessed February 6, 2019]. 2016. URL: `https://ifr.org/service-robots/`.

[89] Omron Robotics and Safety Technologies. *Omron*. [Online; accessed April 25, 2019]. 2019. URL: `https://www.adept.com`.

[90] Stuart J Russell and Peter Norvig. *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited, 2016.

[91] Junji Satake, Masaya Chiba, and Jun Miura. "Visual person identification using a distance-dependent appearance model for a person following robot." In: *International Journal of Automation and Computing* 10.5 (2013), pp. 438–446.

[92] Jürgen Schmidhuber. "Deep learning in neural networks: An overview." In: *Neural Networks* 61 (2015), pp. 85–117.

[93] Florian Schroff, Dmitry Kalenichenko, and James Philbin. "Facenet: A unified embedding for face recognition and clustering." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 815–823.

[94] William Robson Schwartz and Larry S Davis. "Learning discriminative appearance-based models using partial least squares." In: *2009 XXII Brazilian Symposium on Computer Graphics and Image Processing*. IEEE. 2009, pp. 322–329.

[95] Hailin Shi et al. "Embedding deep metric for person re-identification: A study against large variations." In: *European Conference on Computer Vision*. Springer. 2016, pp. 732–748.

[96] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." In: *arXiv preprint arXiv:1409.1556* (2014).

[97] Arnold WM Smeulders et al. "Visual tracking: An experimental survey." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36.7 (2014), pp. 1442–1468.

[98] Sarah Stevensonn. *Dangers of Seniors Living Alone*. [Online: accessed April 26, 2019]. 2017. URL: `https://www.aplaceformom.com/blog/2013-4-1-dangers-ofseniors-living-alone/`.

[99] Chi Su et al. "Pose-driven deep convolutional model for person re-identification." In: *2017 IEEE International Conference on Computer Vision (ICCV)*. IEEE. 2017, pp. 3980–3989.

[100] Xinxing Su et al. "k-Reciprocal Harmonious Attention Network for Video-Based Person Re-Identification." In: *IEEE Access* 7 (2019), pp. 22457–22470.

[101] Christian Szegedy et al. "Going deeper with convolutions." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 1–9.

[102] Dapeng Tao et al. "Deep multi-view feature learning for person re-identification." In: *IEEE Transactions on Circuits and Systems for Video Technology* 28.10 (2018), pp. 2657–2666.

[103] Andrea Thomaz, Guy Hoffman, Maya Cakmak, et al. "Computational human-robot interaction." In: *Foundations and Trends® in Robotics* 4.2-3 (2016), pp. 105–223.

[104] Jim Tørresen. *Multimodal Elderly Care Systems (MECS)*. [Online; accessed April 25, 2019]. 2016. URL: `https://www.mn.uio.no/ifi/english/research/projects/mecs/`.

[105] Andre Treptow, Grzegorz Cielniak, and Tom Duckett. "Active people recognition using thermal and grey images on a mobile security robot." In: *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2005, pp. 2103–2108.

[106] Thanh Q Trinh et al. ""Take a seat, please": Approaching and Recognition of Seated Persons by a Mobile Robot." In: *ISR 2018; 50th International Symposium on Robotics*. VDE. 2016, pp. 1–8.

[107] Rahul Rama Varior et al. "A siamese long short-term memory architecture for human re-identification." In: *European Conference on Computer Vision*. Springer. 2016, pp. 135–153.

[108] Alexander Vorndran et al. "How to Always Keep an Eye on the User with a Mobile Robot?" In: *ISR 2018; 50th International Symposium on Robotics*. VDE. 2016, pp. 1–7.

[109] Brian H Wang et al. "Deep Person Re-identification for Probabilistic Data Association in Multiple Pedestrian Tracking." In: *arXiv preprint arXiv:1810.08565* (2018).

[110] Faqiang Wang et al. "Joint learning of single-image and cross-image representations for person re-identification." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 1288–1296.

[111] Gaoang Wang et al. "Exploit the Connectivity: Multi-Object Tracking with TrackletNet." In: *arXiv preprint arXiv:1811.07258* (2018).

[112] Jin Wang et al. "DeepList: Learning Deep Features With Adaptive Listwise Constraint for Person Reidentification." In: *IEEE Transactions on Circuits and Systems for Video Technology* 27.3 (2017), pp. 513–524.

[113] Shengye Wang and Henrik I Christensen. "TritonBot: First Lessons Learned from Deployment of A Long-term Autonomy Tour Guide Robot." In: *2018 27th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*. IEEE. 2018, pp. 158–165.

[114] Taiqing Wang et al. "Person re-identification by discriminative selection in video ranking." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38.12 (2016), pp. 2501–2514.

[115] Yujiang Wang et al. "A real-time and unsupervised face Re-Identification system for Human-Robot Interaction." In: *Pattern Recognition Letters* (2018).

[116] Longhui Wei et al. "Person transfer gan to bridge domain gap for person re-identification." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 79–88.

[117] Tim Wengefeld et al. "May i be your personal coach? bringing together person tracking and visual re-identification on a mobile robot." In: *Proceedings of ISR 2016: 47st International Symposium on Robotics*. VDE. 2016, pp. 1–8.

[118] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. "Simple Online and Realtime Tracking with a Deep Association Metric." In: *2017 IEEE International Conference on Image Processing (ICIP)*. IEEE. 2017, pp. 3645–3649. DOI: 10.1109/ICIP.2017.8296962.

[119] Lin Wu et al. "Where-and-when to look: Deep siamese attention networks for video-based person re-identification." In: *IEEE Transactions on Multimedia* (2018).

[120] Yu Wu et al. "Exploit the unknown gradually: One-shot video-based person re-identification by stepwise learning." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 5177–5186.

[121] Yu Wu et al. "Progressive Learning for Person Re-Identification with One Example." In: *IEEE Transactions on Image Processing* (2019).

[122] Jun Xiang et al. "Multiple target tracking by learning feature representation and distance metric jointly." In: *arXiv preprint arXiv:1802.03252* (2018).

[123] Tong Xiao et al. "End-to-end deep learning for person search." In: *arXiv preprint arXiv:1604.01850* 2 (2016).

[124] Tong Xiao et al. "Learning deep feature representations with domain guided dropout for person re-identification." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 1249–1258.

[125] Shuangjie Xu et al. "Jointly attentive spatial-temporal pooling networks for video-based person re-identification." In: *arXiv preprint arXiv:1708.02286* (2017).

[126] Yuanlu Xu et al. "Person search in a scene by jointly modeling people commonness and person uniqueness." In: *Proceedings of the 22nd ACM International Conference on Multimedia*. ACM. 2014, pp. 937–940.

[127] Mang Ye et al. "Dynamic label graph matching for unsupervised video re-identification." In: *The IEEE International Conference on Computer Vision (ICCV)*. Vol. 6. 2017.

[128] Dong Yi et al. "Deep metric learning for person re-identification." In: *2014 22nd International Conference on Pattern Recognition (ICPR)*. IEEE. 2014, pp. 34–39.

[129] Wojciech Zajdel, Zoran Zivkovic, and BJA Krose. “Keeping track of humans: Have I seen this person before?” In: *Proceedings of the 2005 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2005, pp. 2081–2086.

[130] Hao Zhang, Christopher Reardon, and Lynne E Parker. “Real-time multiple human perception with color-depth cameras on a mobile robot.” In: *IEEE Transactions on Cybernetics* 43.5 (2013), pp. 1429–1441.

[131] Peng Zhang et al. “Long-Term Person Re-identification Using True Motion from Videos.” In: *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE. 2018, pp. 494–502.

[132] Ying Zhang et al. “Sample-specific svm learning for person re-identification.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 1278–1287.

[133] Rui Zhao, Wanli Ouyang, and Xiaogang Wang. “Unsupervised salience learning for person re-identification.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2013, pp. 3586–3593.

[134] Liang Zheng, Yi Yang, and Alexander G Hauptmann. “Person re-identification: Past, present and future.” In: *arXiv preprint arXiv:1610.02984* (2016).

[135] Liang Zheng et al. “Mars: A video benchmark for large-scale person re-identification.” In: *European Conference on Computer Vision*. Springer. 2016, pp. 868–884.

[136] Liang Zheng et al. “Person re-identification in the wild.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 1367–1376.

[137] Liang Zheng et al. “Scalable person re-identification: A benchmark.” In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 1116–1124.

[138] Meng Zheng, Srikrishna Karanam, and Richard J Radke. “RPIfield: A New Dataset for Temporally Evaluating Person Re-Identification.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2018, pp. 1893–1895.

[139] Wei-Shi Zheng, Shaogang Gong, and Tao Xiang. “Associating Groups of People.” In: *British Machine Vision Conference (BMVC)*. 2009.

[140] Wei-Shi Zheng, Shaogang Gong, and Tao Xiang. “Towards open-world person re-identification by one-shot group-based verification.” In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38.3 (2016), pp. 591–606.

[141] Zhedong Zheng, Liang Zheng, and Yi Yang. “A discriminatively learned cnn embedding for person reidentification.” In: *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 14.1 (2018), p. 13.

[142] Zhedong Zheng, Liang Zheng, and Yi Yang. “Unlabeled samples generated by gan improve the person re-identification baseline in vitro.” In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 3754–3762.

[143] Jianqing Zhu et al. “Deep hybrid similarity learning for person re-identification.” In: *IEEE Transactions on Circuits and Systems for Video Technology* 28.11 (2018), pp. 3183–3193.

[144] Xiatian Zhu et al. “Fast open-world person re-identification.” In: *IEEE Transactions on Image Processing* 27.5 (2018), pp. 2286–2300.