POLITECNICO DI TORINO

Master's Degree in Data Science and Engineering

Master's Degree Thesis

Synthetic-to-Real Domain Transfer with Joint Image Translation and Discriminative Learning for Pedestrian Re-Identification

Supervisors

Prof. Barbara CAPUTO

Dott. Mirko ZAFFARONI

Candidate

Antonio Dimitris DEFONTE

July 2022

Abstract

Person re-identification is a challenging computer vision task where one wants to match each probe pedestrian to the corresponding images in the gallery set. Pose, viewpoint and illumination variations have been well-known issues. Despite this, recent developments have shown positive results when models are trained and tested on the same dataset. However, different datasets present unrelatable characteristics, to the point that they define distinct domains. So far, achieving a good performance on cross-domain approaches has been proven to be much more demanding than training standard supervised methods. Recent models that bridge the gap across domains have drawn significant attention since, from a practical perspective, annotating new data is error-prone and time-consuming, whereas having unlabeled images is much less expensive. Moreover, the emerging field of synthetic pedestrian re-identification is gaining momentum. Instead of employing real world-data, the environments are computer-generated. On top of easing the annotation process, this gives more freedom relative to what is available in a real-world scene. From another perspective, synthetic data also addresses ethical issues such as recording people without authorization and exploiting those videos for sensitive applications.

The objective of this work was to generalize from our synthetic dataset GTASynthReid, exclusively built by exploiting the graphic engine of *Grand Theft Auto V*, to real-world data. Starting from these motivations, we embraced a generative approach that performs synth-to-real image translation and jointly learns pedestrian feature descriptors. We injected target domain information into a network trained on the source identities. To the best of our knowledge, we are the first to adopt the *Contrastive Unpaired Translation* framework in our task. Instead of learning via "cycle consistency", it encourages corresponding patches of the input and output images to be similar, allowing "one-way" translation. We also designed a feature matching loss for the discriminator to increase performance. We show that, although current methods obtain scores that are difficult to reach, our pipeline can achieve results that are comparable to and even better than earlier similar approaches, both with real and synthetic data. We also show that the similarity between our dataset and each target increases after the image translation.

Acknowledgements

I would like to express my most heartfelt thanks to everybody who supported me in these recent difficult times.

I will start by expressing my deepest gratitude to Dott. Mirko Zaffaroni who supported me with his knowledge and insights, guiding me both during this thesis and the internship.

I want to thank Dott. Manuel Scurti who helped me to find meaningful resources during my internship.

I am grateful to professor Barbara Caputo and the Links Foundation for making this project possible and allowing me to work on such exciting and advanced topics.

I express my utmost gratitude to all my friends and colleagues that emotionally supported and helped me during these years, this experience would have been much more difficult without them. I am sure you will succeed in everything you put your mind on. I wish you good luck in your future endeavors.

I finally want to thank my family that financially supported me during my attendance at Politecnico di Torino.

Table of Contents

List of Tables

List of Figures

Acronyms

ADAM
Adam Optimizer

AI
Artificial Intelligence

BoT
Bag of Tricks

CBIR
Content-Based Image Retrieval

CBN
Camera-based Batch Normalization

CMC
Cumulative Matching Characteristics

CNN
Convolutional Neural Network

CUT
Contrastive Unsupervised Learning

DPM
Deformable Part Model

FID
Fréchet Inception Distance

GAN

Generative Adversarial Network

GPU

Graphics Processing Unit

GTAV

Grand Theft Auto V

IS

Inception Score

JVTC

Joint Visual and Temporal Consistency

MaP

Mean Average Precision

MLP

Multilayer Perceptron

MMD

Maximum Mean Discrepancy

MSE

Mean Squared Error

PCB

Part-based Convolutional Baseline

Re-ID

Re-Identification

RPP

Refined Part Pooling

SGD

Stochastic Gradient Descent

SVM

Support Vector Machines

UDA

Unsupervised Domain Adaptation

Chapter 1

Introduction

In this thesis, we developed a framework for adapting our synthetic dataset to other real-world ones in the context of *pedestrian re-identification*. This work is intended to complete the internship where I developed the synthetic dataset *GTASynthReid* by exploiting only the graphic engine of the video game *Grand Theft Auto V*. Both the internship and this thesis were carried out at the *Links Foundation* in the *Data Science For Industrial & Societal Applications* area. The Links Foundation operates at a national and international level on topics such as industry 4.0, smart mobility, agritech, space economy, security and intelligence. It is an effort to promote digital innovation and increase competition by acting as a bridge between academic research and the workplace. In this introductory chapter we will first provide a brief overview of the challenges around person re-identification and then state our objectives and contributions.

1.1 Pedestrian re-identification

This task concerns recognizing pedestrians recorded in different overlapping or non-overlapping videos where pose, illumination, viewpoint, and environments vary. It has practical applications in surveillance and security since person re-identification systems can provide additional intelligence to monitor high-risk areas. Such systems often include multiple recording cameras monitoring a target pedestrian in different areas. Typically, for each camera, we first need to extract, frame by frame (when feasible), the bounding boxes around each pedestrian. Then, given a pedestrian picture from one camera view, one needs to match it against those extracted from other viewpoints. For a successful retrieval, it is fundamental to encode good pedestrian descriptors, which at this point is almost always done with neural networks.

1.1.1 Standard setting

In its simplest form, we evaluate person re-identification models on the same dataset employed for the training and feature extraction. Each dataset has a training partition used for supervised training with known identities and a test partition used for the retrieval. It is crucial to notice that these two partitions do not share the same pedestrian identities. The test partition is further divided into query and gallery sets. In this sense, after training we will employ the neural network as a feature extractor to encode each image from the query and gallery sets. Then, given an encoded probe identity from the query, we will compute some distance between the probe itself and the images in the gallery, obtaining a ranking with the most similar pedestrians at its top. One hopes that the images sharing the probe identity will have a higher rank.

1.1.2 Cross-domain

Albeit interesting, what we just explained is often not practical for real scenarios. Models have difficulties in generalizing to unseen viewpoints, illumination and other domain-specific conditions, meaning that one will see a huge drop in performance when testing a model on a different dataset. On top of this, manually annotating large quantities of data for each domain is error-prone, time-consuming and impractical. In this direction, there have been recent advances in *cross-domain person re-identification*, usually in the form of *unsupervised domain adaptation*. To this end, one needs to employ two datasets: the source and the target. We wish to achieve a good performance on the target test partition by adapting the labeled source data to the unlabeled training partition of the target dataset. The target test identities remain unseen during the training process. In a real-world scenario, this would translate into having availability of non-annotated images in a specific target environment, which is much simpler than obtaining labeled data.

In other unsupervised domain adaptation tasks, it is common to adapt from the source to the target while validating on another dataset. Regarding our task instead, this procedure is quite uncommon. There are predefined source-to-target performance evaluation configurations to which one can compare a model. However, there exist works that approach domain generalization techniques training and testing on several datasets.

1.1.3 Synthetic data

Software-generated datasets have gained momentum in the computer vision community. Recently, virtual worlds are a growing interest also in the sub-field of person re-identification. This kind of data relieves the burden of manually labeling large quantities of images, allowing more controllable environments for the researchers'

needs. Another issue concerns ethics. For real-world datasets, one will have to record several pedestrians from multiple points of view. This is usually done without asking for the pedestrians' consent, despite of existing regulations. Synthetic data also alleviates this problem. However, we now have to adapt our synthetic dataset to real environments. Depending on characteristics like the learning paradigm of choice, the quality of the computer-generated pedestrians and the gap between the real and synth domains, this can be a more or less arduous task. In this scenario, besides the standard evaluation of each synth-to-real model for the predefined target, we also evaluated on the other real datasets, resembling a more general synth-to-real adaptation.

1.2 Contributions and work structure

There are many ways to address this task. We decided to adopt a generative approach for image translation that jointly learns discriminative pedestrian descriptors. We will now state this thesis objective and list our contributions. Finally, we provide an outline for the rest of the manuscript.

1.2.1 Objective

The objective of this thesis was to develop a deep learning framework for pedestrian re-identification to bridge the gap between our synthetic dataset GTASynthReid and other real-world datasets. We evaluated our model on Market1501 [1], DukeMTMC-Reid [2] and CUHK03 [3]. The dataset was developed during the internship by solely exploiting the graphic engine of Grand Theft Auto V.

1.2.2 Contributions

We embraced a generative approach that jointly translates images from the synth to the real domains and injects target domain-specific information into a *Resnet-based* [4] network trained in a supervised fashion on the source identities. Our contributions are as follows:

- we extended a Pytorch framework[1] [5] for person re-identification so that it can also accommodate learning via translation from synthetic data;
- to the best of our knowledge, we are the first to employ [6] for pedestrian re-identification. This contrastive unpaired translation framework avoids the

[1]The original work can be found at https://github.com/KaiyangZhou/deep-person-reid.

cycle consistency loss [7] and double architecture structure common to many other methods;

- we designed a similarity loss inspired from [8] that matches features at different discriminator layers when the discriminator itself is fed with a pair of source and translated images or a pair of target and translated images (when switching the perspective from the discriminator's point of view to the generator's point of view);
- we were able to achieve a better performance compared to a Resnet baseline trained for direct transfer;
- although we could not surpass the current state of the art (given also the limitations of our dataset), we achieved better results than some earlier real-to-real adaptation methods with a lighter, one-stage synth-to-real model. We also measured the distance by means of FID [9] between the real and our GTASynthReid datasets before and after the translation, observing a lower value after the translation.

1.2.3 Outline

We structure this thesis into chapters that progressively build and complete the knowledge required to understand our work. Starting from the deep learning fundamentals, we will then explain how pedestrian re-identification works, also across domains. After that, one can understand our implementation details and results. The remaining chapters are as follows.

2. *Neural Networks.* In this introductory chapter, we explain the general concepts of machine and deep learning and show the theoretical foundations of modern deep learning.
3. *Pedestrian Re-Identification.* Here we introduce the person re-identification task, showing existing datasets, methods and metrics for performance evaluation.
4. *Cross-Domain Transfer.* After a brief introduction to general domain adaptation techniques, this chapter completes the previous one by showing the current state of the art for person re-identification across domains.
5. *Model Architecture and Experiments.* In this chapter we show the implementation of our model and the obtained results, comparing it with existing works.
6. *Conclusions.* We devote the last chapter for remarks and possible future directions.

Chapter 2

Neural Networks

This chapter introduces the foundations of modern neural networks. Before getting into the details of *pedestrian re-identification* (Re-ID), it is essential to study the components of such architectures in a simplified environment. We will mostly refer to classification and expand eventually to broader concepts. After a brief presentation on the origins of artificial intelligence (AI), we describe the modules of generic fully connected and convolutional neural networks. Having explained the differences between normalization, pooling and fully connected layers, it is crucial to understand how to measure the goodness of the current network weights configuration. To do so, we explain the theory behind some loss functions that will also be useful in later chapters. Then, we show how the learning process actually works by means of optimization. After having reviewed all these basic building blocks, we will introduce and motivate some well-known convolutional neural networks (CNNs) architectures. To conclude the chapter, we will mention a different learning paradigm, generative learning. Since this work greatly relies on generative adversarial neural networks (GANs), we will only focus on those. In the next two chapters, we will provide more details on how to use such architectures in the context of person Re-ID.

2.1 Some historical remarks

The ability to develop artificial agents capable of intelligent human behavior has always fascinated humanity. Pioneers of AI tried to model simplified versions of our brain to autonomously solve generic tasks without having to write a specific computer program each time. As of today, biological models are not the main source of inspiration since the field of AI is deeply being influenced by disciplines such as linear algebra, probability and statistics, optimization and computer science. We are still a long way from building artificial general intelligence systems. However,

we can develop autonomous processes for image classification, text translation, image-to-text and much more. For some tasks, such models can even surpass human performance. Although it is difficult to provide an exact date for the birth of AI, many authors and researchers point to the *Dartmouth Workshop* [10] in 1956. This conference was organized by distinguished engineers, mathematicians and psychologists such as John McCarthy, Marvin L. Minsky, Nathaniel Rochester and Claude E. Shannon. Having said that, the evolution of AI has not been linear. The initial high expectations had to come to terms with the limited computing power of those days, which led to criticism and slowed the development of this field (AI winters). If we consider the subfield of deep learning and neural networks alone, it only gained momentum starting from the first decade of this century, when better and cheaper graphics processing units (GPUs) became available.

In the next sections we will explain some key differences between machine and deep learning, without focusing on artificial general intelligence. With this in mind, we quote from T. Mitchell [11] for what could be a broad definition of an autonomous learning process:

> *A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.*

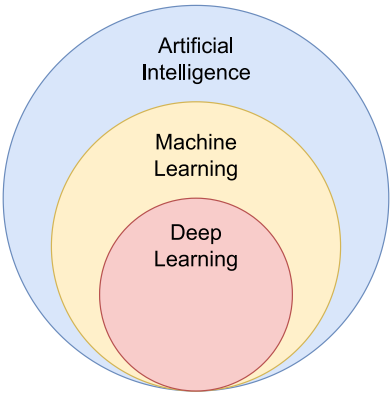

Figure 2.1: One of the possible interpretations of the relationships between artificial intelligence, machine and deep learning. In this context, AI includes even artificial general intelligence, while machine learning refers to all those techniques and problems for which it is possible to learn from experience (classification, regression, unsupervised learning, *etc...*). Deep learning, instead, is the subset of techniques that address the same problems of machine learning but with deep architectures.

2.1.1 Machine learning

There are many definitions of machine and deep learning, often distinct from statistical learning [12]. For the scope of this thesis, we refer to deep learning as a subset of machine learning (as in figure 2.1), where, instead of employing shallow architectures, problems are solved with deep (more than one *layer*) neural networks. In the machine learning theory, a classic example is the halfspace problem [12]. Given a set of linearly separable data $\boldsymbol{X}$ (*i.e.*, there exist a separating hyperplane) with binary labels $\boldsymbol{Y} = \{-1, +1\}$, the goal is to find a linear separator such that each datapoint lies on the correct side (depending on its label). We can express this with the following homogeneous decision function: $\text{sign}(\langle \mathbf{w}, \mathbf{x} \rangle)$, where $\mathbf{x} \in \boldsymbol{X}$ and $\mathbf{w}$ is the normal to the separating hyperplane we seek to find. In 1958, Frank Rosenblatt developed a simplified model of the biological neuron, the *perceptron* algorithm [13]. His procedure, starting from an initial $\mathbf{w}$, will iteratively correct the elements of $\mathbf{w}$ itself so that, for all $(\mathbf{x}_i, y_i)$, where N is the cardinality of the dataset, we have $y_i \langle \mathbf{w}, \mathbf{x}_i \rangle > 0$. This is equivalent to say that all points lie on the correct side of the hyperplane parameterized by $\mathbf{w}$. If, for the current configuration of $\mathbf{x}$, there are datapoints on the wrong side, *i.e.* $y_i \langle \mathbf{w}, \mathbf{x}_i \rangle \leq 0$, we update $\mathbf{w}$ for the next iteration to be $\mathbf{w} + y_i \mathbf{x}_i$. For instance, suppose that the values of $\mathbf{w}$ are such that they classify as -1 a sample $\mathbf{x}$ with $y = +1$. In this case, the perceptron algorithm will add to $\mathbf{w}$ the sample $\mathbf{x}$. By increasing the value in each entry of $\mathbf{w}$, we hope that, in the next iteration, this datapoint will be correctly classified. Algorithm 2.1 shows the entire procedure.

In the previous context, what we have just explained finds one of the (infinite) existing separating hyperplanes. One might argue whether one particular separator is better than the others. Suppose that, for a given set of data, we found a hyperplane far closer to the datapoints of the positive class. If we sample new data, we might end up with errors, since some examples of the positive class could cross the separator. In this sense, one could try to find the hyperplane that better separates the data, *i.e.*, the separator that maximizes the minimum distance (known as margin) between the points and itself. To do so, we can solve the halfspace problem with the *support vector machine* (SVM) algorithm. However, we are still assuming the linear separability of the data. To relax this assumption, one can employ the *soft*-SVM, which allows for mistakes (points of wrong class crossing the margin). Nonetheless, real-world data (*e.g.*, sensors, DNA sequences, images, *etc...*) often lives in high-dimensional convoluted manifolds. In these scenarios, finding a linear separator in the original space often leads to poor results. One could map the data into a higher dimensional space, "unfolding" the manifold and finding a linear separator. Despite this, mapping to a higher dimensional spaces brings all the problems related to the *curse of dimensionality*. We can avoid this issue by using *kernels*. The only concept we care about in that space is the inner product

between our data. Consequently, instead of mapping to higher dimensions, we adopt a function that implements the inner product in the high-dimensional space (kernel trick). We transformed the original problem into a parametric problem where the weights are inner products of our data (there are some conditions to satisfy). However, with this new formulation, if the cardinality of our data is huge, we will still end up in a high-dimensionality scenario. A related concern is the pre-processing of the data. Machine learning pipelines usually rely on handcrafted features. This process ranges from simply aggregating different features, to reducing the dimensionality and extracting relevant data descriptors (*e.g*: sift [14], surf [15] for images). This requires a high amount of human supervision and the overall results could strongly depend on on the chosen descriptor. These are among the reasons why deep learning methods are so successful since we often have high availability of data and the feature crafting process is performed automatically. In the next section, we will expand more on this topic.

Algorithm 2.1 Perceptron algorithm [13] for the binary halfspace problem [12] when the datapoints are linearly separable.

procedure PERCEPTRON($\boldsymbol{X}, \boldsymbol{Y}, \mathbf{w}$)
    ▷ $\mathbf{x} \in \boldsymbol{X}$ are the datapoints while $y \in \boldsymbol{Y} = \{-1, +1\}$ the labels
    ▷ $(\mathbf{x}_i, y_i)$ is a tuple with a datapoint and its label
    ▷ $i \in \{1, 2, ..., N\}$, where N is the number of datapoints
    ▷ $\mathbf{w}$ is the vector normal to the hyperplane we seek to find.
    ▷ Initialization
    $t \leftarrow 0$ ▷ Iteration counter set to 0
    $\mathbf{w}(t) \leftarrow \{0, 0, ..., 0\}$ ▷ Normal vector initialization
    ▷ Execution
    while $\mathbf{w}$ not converged **do**
        if $(\exists\, i \mid y_i \langle \mathbf{w}(t), \mathbf{x} \rangle \leq 0)$ **then**
            $\mathbf{w}(t+1) = \mathbf{w}(t) + y_i \mathbf{x}_i$ ▷ Update normal vector
        end if
        $t \leftarrow t + 1$ ▷ Update iteration counter
    end while
    return $\mathbf{w}(t)$ ▷ Return correct normal vector
end procedure

2.1.2 Learning paradigms

In the previous subsection, we mentioned a binary classification example, the halfspace problem. In that scenario, the goal was to learn a hyperplane such that each sample lies on the correct side. To do so, we exploited the labels coupled

with the datapoints. One might ask if there are different learning paradigms. As explained in [16], depending on the label annotation availability, we might end up in different learning settings.

- *Supervised learning.* In this learning paradigm each datapoint is associated to a label representing its category. We can use the available data and labels to train a model and then predict the classes of unseen data. This task is known as *classification*. An alternative is predictive *regression*, where, instead of having labels, one needs to exploit and predict continuous values (*e.g.*, predict the price).

- *Unsupervised learning.* In this setting, we do not have the availability of hard-coded labels. There are different schools of thought on what can be considered unsupervised learning, with some researchers limiting to *clustering* methods while others allowing procedures that employ some level of ground-truth, such as *pseudo labels*. However, it does not involve human annotation.

- *Self-supervised learning.* This kind of learning paradigm makes use of pseudo labels that are usually generated from data properties. It is often adopted for pretext tasks alongside the downstream one. Since it does not employ human annotation, it is considered a subset of unsupervised learning by some authors [16].

- *Semi-supervised learning.* Another common scenario is characterized by the availability of a small labeled subset of data. The majority of the datapoints are instead unlabeled, bringing to methods that consider multiple approaches.

These are only the learning paradigms that are interesting for our work. In reality, there exist other ones that we will not consider, such as *reinforcement learning* and *weakly supervised learning*. Regarding unsupervised learning, we already mentioned that some authors might include more approaches. In chapter 4, we will make use of the term *unsupervised domain adaptation* (as in the literature). This refers to techniques that do not employ target data labels, even if the learning process is supervised by, for instance, pseudo labels or labeled source data.

2.1.3 Training pipeline

We will now explain how to train a generic model. These steps can be followed in both machine and deep learning pipelines. Intuitively, one would like to generalize well to unseen data, achieving a low error measured by some function (see section 2.3). However, if we were to train a model on the whole available dataset, we could risk to fit too well (*i.e.*, overfit) only on our available data, performing poorly on unseen samples. For this reason, the original dataset is usually splitted into

training set, on which we train our model, and test set, where we evaluate our model. Additionally, the training pipelines often require hyperparameters tuning, meaning that one has to experiment with multiple sets of controllable parameters to find the best performing model. Consequently, one has to hold out from the training data a smaller subset of samples, the validation set. To summarize, we will first learn and tune a model on the training set to measure its performance on the validation set. Then, we will retrain the model with the best set of hyperparameters on the training and validation data. Finally, we can measure the performance on the test set. A better validation procedure divides the training set into k parts, iteratively training on $k-1$ partitions and validating on the remaining one. This compensates for the overestimation of the test error that a standard training-validation split might bring to since we would train on fewer observations [17]. Known as k-fold cross-validation, this process is not always feasible, particularly with deep learning models. On top of this, as we will mention in chapter 4, in some scenarios is more challenging to design a validation set, being sometimes a research topic.

2.2 Neural networks structure

One could think of neural networks as function approximators. As explained in [18], suppose that there exists a function f^* that maps the input data to the correct category, *e.g*: $y = f^*(\mathbf{x})$. If we express a neural network model as a function f parameterized by $\boldsymbol{\theta}$, it will learn the values of its parameters such that the mapping $y = f(\mathbf{x}, \boldsymbol{\theta})$ best approximates the original one. This can be extended beyond classification. Another interpretation considers all the layers of a neural network, but the last one, as automatically generated data features. Instead of manually designing strong feature descriptors, each layer encodes some kind of information, ranging from more general to task-specific concepts. The last layer can be thought of as a classifier that works with the learned feature maps. All the architectures that we will explain are labeled as *deep feedforward neural networks*, meaning that the data flows into the layers without cycles in the computational graph. Architectures that involve loops fall under the realm of *recurrent neural networks* (out of the scope of this work).

2.2.1 Fully connected neural networks

Fully connected neural networks are characterized by what is known as *fully connected layer*. These networks known also as *multilayer percetrons* (MLPs) because of their similarity. The main difference is that now we have at least three layers: an input, an output and a hidden layer. The number of hidden layers is a hyperparameter to be optimized. Each layer can have one or more units (*neurons*). For each unit, we compute the linear combination (thus the name fully connected)

of the previous layer outputs, similar to what we did in the perceptron model. If $\mathbf{o}$ is the output of the previous layer and $\mathbf{w}$ are the weights associated to a neuron, we have that $\sum_{i=1}^{M} \mathbf{w}_i \mathbf{o}_i$ will be the output of that neuron. The result of all the layer units will constitute the layer output. Figure 2.2 provides a visual explanation of this process. Notice that the number of units in a layer is commonly known as channel size.

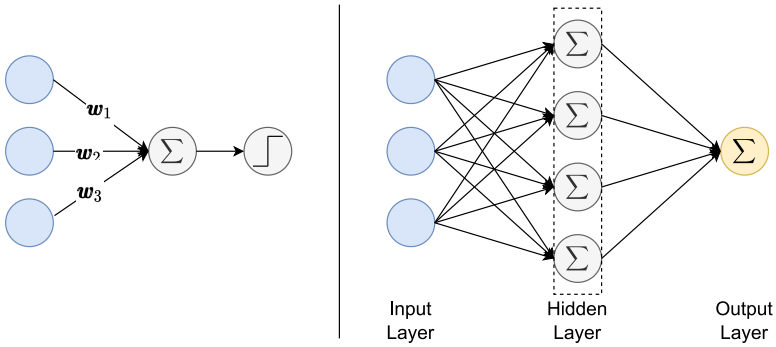

Figure 2.2: On the left-hand side, we have a standard perceptron with a threshold function (sign function as we described before). On the right-hand side, there is instead a MLP with a hidden layer made of four neurons. Each neuron has its set of weights, both for the hidden and output layers (we omitted the activation functions and weights for clarity). Each unit of the input layer is relative to a feature of the original data.

2.2.2 Convolutional neural networks

If we were to use fully connected layers with images, we would need to connect each input pixel to each unit of the first hidden layer. However, this would prevent us from exploiting an important property of images. Generally speaking, neighboring pixels can present strong correlations, meaning they could belong to the same object of interest as a leg, hand, dog or person. To take advantage of these spatial correlations, we need to introduce the convolution operation. In practice, we slide the input image with small filters (also known as *kernels*), summing the elements-wise product each time we change region (see figure 2.3). The sliding stride is a hyperparameter that tells us by how many spatial location we should move after each operation. Typical kernel sizes are 3×3, 5×5 and 7×7, but one could adopt also rectangular filters (*e.g.*, 1×3, 3×1, *etc*...). If needed, it is possible to add padding to the image. Each kernel has a depth size equal to the channel size of the previous layer output. For instance, considering RBG images, the first convolutional layer will have units of three two-dimensional filters, one for each input channel. The results of each channel are then summed together. The

number of distinct filters (considering each three-dimensional filter as a single unit) represents instead the current layer channel size. Architectures that employ this kind of layer are known as convolutional neural networks (CNNs).

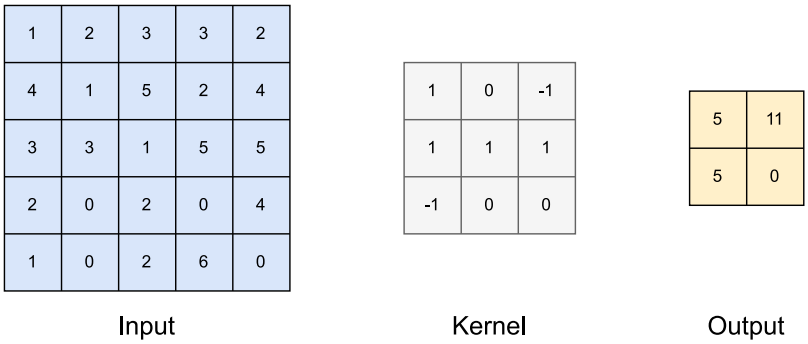

Figure 2.3: Illustration of the convolution operation. We slide the gray kernel over the input with stride 2, obtaining the yellow result.

2.2.3 Activation functions

Neural networks have high expressive power, being able to make up for the data nonlinearities. However, simply stacking convolutional or fully connected layers does not involve any nonlinear operations. For instance, suppose that the first hidden layer of a neural network has its units parameterized by the matrix $\boldsymbol{W}_1$, while the second one is parameterized by $\boldsymbol{W}_2$. Given an input $\mathbf{x}$, the raw output after the second layer is $\boldsymbol{W}_2(\boldsymbol{W}_1\mathbf{x})$. This is equivalent to $\boldsymbol{W}\mathbf{x}$, where $\boldsymbol{W} = \boldsymbol{W}_1 \times \boldsymbol{W}_2$. By definition, this is a linear operation that does not enable a neural network to approximate nonlinear functions. If f represents the sigmoid function, we can rewrite the previous function as a nonlinear one by passing the outputs of each layer to f, that is $f(\boldsymbol{W}_2 f(\boldsymbol{W}_1\mathbf{x}))$. In the deep learning context, these nonlinearities are called activation functions. Below we briefly describe some of the existing ones.

- *Sigmoid.* The sigmoid function has the characteristic "S-shaped" curve. It bounds its input to the range $(0, 1)$ and it can be defined as $\sigma(\mathbf{x}) = \frac{1}{1+e^{-\mathbf{x}}}$. However, this function has two main problems. Because of its flat regions, it will "kill" the gradients during the backpropagation (see section 2.4) when the input is far from zero. Another issue is that the output will not be centered at zero. This translates into having a gradient always positive or negative, depending on the input sign. Updating the gradients in the same direction will generally require more steps, leading to inefficient optimization.
- *Tanh.* This activation function will bound the output of a neuron to the range (-1, 1). As its name suggests, it is computed as the tanh of its input. It solves

the zero-centering problem, but the vanishing gradient issue is still present as for the sigmoid function (flat regions).

- *ReLU.* The *rectified linear unit* (ReLU) [19] is among the most employed activation functions in modern neural network architectures. It can be written as $\sigma(\mathbf{x}) = \max(0, \mathbf{x})$. Contrary to the two previous nonlinearities, it does not upper bound its input, preventing saturation and dead gradients in the positive part. However, when $\mathbf{x} < 0$, we still have the vanishing gradient problem (although there are techniques that help to avoid this) and the output is not zero-centered.

- *Leaky ReLU.* This activation function has all the nice properties of the ReLU and, on top of this, prevents dead gradients in the negative part. This nonlinearity can be expressed as $\sigma(\mathbf{x}) = \max(0.01\mathbf{x}, \mathbf{x})$ or $\sigma(\mathbf{x}) = \max(\alpha\mathbf{x}, \mathbf{x})$, where we practically allow a non-zero slope when $\mathbf{x} < 0$. This can be either predefined or learned as a parameter (α in the second version). By doing so, there are no flat regions and the output is more centered.

One will notice that the most adopted activation functions belong to the family of ReLUs, because of the aforesaid properties. For the zero-centering problem, as we will see later (subsection 2.2.6), there exist additional layers that normalize the nonlinearity inputs. We provide a visual explanation of the mentioned activations in figure 2.4.

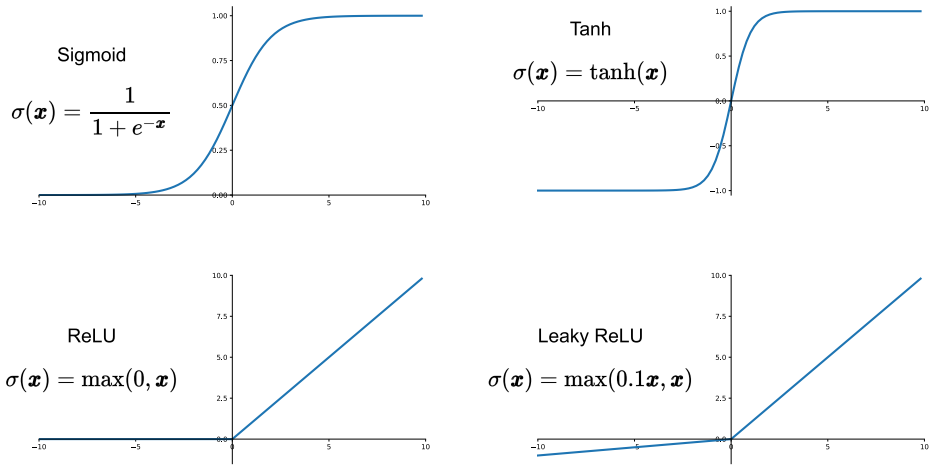

Figure 2.4: Illustration of the sigmoid, tanh, ReLU and leaky ReLu activation functions. In the leaky ReLU we multiply by 0.1 to emphasize the slope in the negative region.

2.2.4 Pooling layer

After the activation layer, one will usually find a pooling operation. This preservers the channel dimension of the output feature maps while downsampling the spatial dimensions. The layer applies a predefined operation that will result in some summary statistics. We can do this by computing the statistic over the spatial locations inside a sliding window of a certain size and stride. The outcome will render the feature representation approximately invariant to local (small) input translations [18]. Suppose that one wants to detect whether an object is present in the input image. Given that we do not care about its exact location, even if the object of interest slightly moves, most of the outputs of the pooling layer will roughly be the same. Among others, average pooling and max pooling are widely used, with variants such as global average pooling that summarizes all the spatial locations into one value. Global pooling often replaces fully connected layers in CNNs, maintaining spatial information and being more meaningful as it relates feature maps to class categories [20].

2.2.5 Dropout layer

Fully connected layers have a high amounts of learnable parameters that could expose a model to overfitting risks. This happens in the form of co-adaptation, meaning that a model correlates the categories to the input samples rather than the learned features. The result is a poor generalization capability. An option could be to use fewer parameters, reducing the model capacity. The dropout layer [21] aims at solving this problem (see figure 2.5 for an illustration). During training, one has to randomly disconnect (usually with a probability of 0.5) some of the dense links. The model is now forced to learn from different features and cannot just memorize to use a specific set of them. During test time instead, we reactivate the connections using all the available features.

2.2.6 Normalization

As we showed before, having normalized data is important during training, speeding up the convergence and avoiding vanishing gradients. One could center the data as a pre-processing step. Still, this is not enough. After each convolutional or fully connected layer, the result is not normalized anymore. Work [22] provides a solution to this issue, batch normalization. They add a layer before (other works add it after) the nonlinearity, normalizing each batch with its statistics. During training, for each batch of data $\mathcal{B} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_m\}$, we compute the following

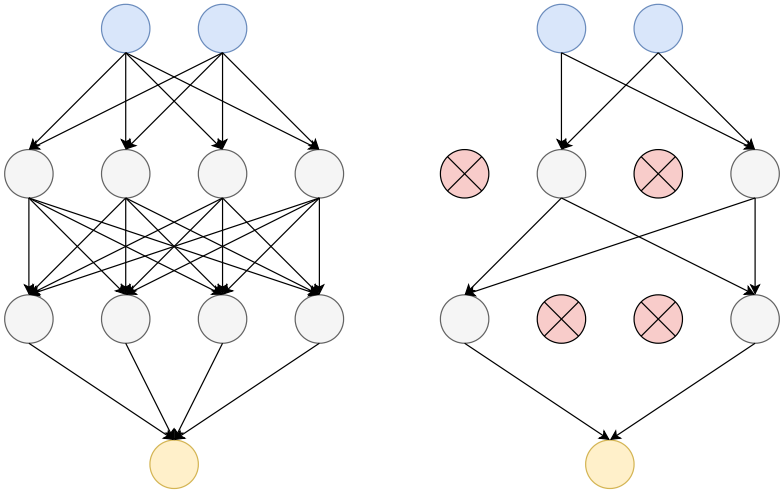

Figure 2.5: Dropout example with dummy network. Inputs are in blue, hidden layers in gray and output in yellow. On the right hand side, we disconnected the links of the red units.

statistics:

$$\begin{aligned} \mu_B &= \frac{1}{m}\sum_{i=1}^{m}\mathbf{x}_i \quad \sigma_B = \frac{1}{m}\sum_{i=1}^{m}(\mathbf{x}_i - \mu_B)^2 \\ \hat{\mathbf{x}}_i &= \frac{\mathbf{x}_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \quad \forall \mathbf{x}_i \in \mathcal{B} \\ \mathbf{y}_i &= \gamma\hat{\mathbf{x}}_i + \beta \quad \forall i \in \{1, 2, ..., m\} \end{aligned} \tag{2.1}$$

where μ_B, σ_B are respectively the batch mean and standard deviation while $\hat{\mathbf{x}}_i$ is a datapoint and ϵ is a small constant for numerical stability. In this sense, we are estimating the mean and variance of each activation [22] (for convolutional layers we estimate different statistics for each channel). On top of the normalization operation, they also add the possibility to learn the identity transformation. The motivation is that normalizing could alter the layer representation or expressive power. For this purpose, to obtain $\mathbf{y}_i$, this method uses two learnable parameters, γ and β. If $\gamma = \sqrt{\sigma_B^2}$ and $\beta = \mu_B$, then $\mathbf{y}_i$ would rematch $\mathbf{x}_i$. For the inference stage instead, one can track during training, for each layer, some kind of moving average for the mean and variance over the different batches of data. After the training process, we will have an estimate of the mean and standard deviation relative to each layer. We can then apply those statistics to the test data, using the same γ and β learned during training.

Having said that, there exist also other kinds of normalization. An example is instance normalization [23] that computes statistics for each sample in the batch (for convolutional layers it estimates different statistics for each channel of each

instance). The procedure is usually repeated also at test time. This normalization strategy was introduced for image transfer methods (see chapter 4) since the original authors obtained significant improvements in the image generation. Work [24] explains how batch normalization preserves discriminative power among samples while instance normalization separates instance contrast information from the content.

2.3 Loss functions

Given a parameters configuration of a model, one would like to evaluate how much is paying because of using that configuration. In simpler terms, we can employ some functions that generally accept the output of a model (given a batch of data) and a ground truth. They then compare their arguments and retrieve a real number representing how well the model is performing. These functions are known as *loss functions* and deeply relate with statistical concepts such as maximum likelihood estimation and information theory.

2.3.1 Cross-entropy loss

When dealing with classification problems, one will generally use the well-known *cross-entropy loss*. Supposing that our dataset has C classes, the model of choice $f(\mathbf{x}, \boldsymbol{\theta})$ will output from its classifier (last fully connected layer) the logit vector $\mathbf{o}$ of C raw, non-normalized predictions. The *softmax function* translates them into the probability that the input data $\mathbf{x}$ will be assigned to a certain class k :

$$P(c_k = 1 \mid \mathbf{x}) = \frac{e^{\mathbf{o}_k}}{\sum_{j=1}^{C} e^{\mathbf{o}_j}} \tag{2.2}$$

where each entry of the logits will be mapped to the interval $(0, 1)$. Assuming that each input data $\mathbf{x}$ is associated to a one-hot encoded ground truth vector $\mathbf{y}$, we can express this loss as:

$$\mathcal{L}_{Class} = \mathbb{E}_{\mathbf{x}\sim\mathcal{B}}\Big[\sum_{i=1}^{C} \mathbf{y}_i \log(P(c_i \mid \mathbf{x}))\Big] \tag{2.3}$$

where $\mathbf{y}$ is zero everywhere but in the k^{th} position of the correct category in which it is one and $P(c_i \mid \mathbf{x})$ is as in eq. 2.2. Other than averaging over the batch $\mathcal{B}$ of data, there are alternative reduction methods (*e.g.*, just the sum).

2.3.2 Mean squared error

Instead of predicting classes, one might be interested in regression. In such scenarios, the neural network outputs a single real value that will be compared to the input

data ground-truth. This can be extended to compare also output vectors or feature maps (as explained in chapters 4 and 5). There are multiple loss functions for this task, we present the *mean squared error* (MSE) loss, also known as *L2 loss*. It can be written as:

$$\mathcal{L}_{MSE} = \mathbb{E}_{\mathbf{x}\sim\mathcal{B}}\Big[(y - f(\mathbf{x}, \boldsymbol{\theta}))^2\Big] \tag{2.4}$$

where, for simplicity, we compare the output of a model f with the real ground-truth value y associated to the input $\mathbf{x}$. As in the previous loss function, there are other reduction methods besides averaging over the batch of data $\mathcal{B}$.

2.3.3 Contrastive and triplet losses

In learning paradigms such as one-shot, few-shot learning and image retrieval, deep metric techniques are quite common. In these settings, one would like to learn a function that maps semantically similar points from the original data space, to metrically similar points in the embedding space (likewise for semantically different points) [25]. This function can be approximated by a neural network which is guided by loss functions that push together, in the embedding space, samples that belong to the same category and pull apart otherwise. For instance, *siamese* [26] neural networks work with pair of images, often adopting the contrastive loss [27]:

$$\mathcal{L}_{Contr} = \frac{1}{2} Y D^2 + \frac{1}{2}(1 - Y)(\max(0, m - D))^2 \tag{2.5}$$

where $D = \|f(\mathbf{x}_1, \boldsymbol{\theta}) - f(\mathbf{x}_2, \boldsymbol{\theta})\|_2$ is the euclidean distance in the embedding space of two input samples encoded by the neural network f parameterized by $\boldsymbol{\theta}$. The binary label Y is one if the inputs are similar, zero otherwise. The margin term $m > 0$ is fundamental to avoid trivial solutions. As explained in [27], it practically defines a radius around the embedding discarding dissimilar pairs contribution when their distance exceeds that radius.

The *triplet loss* [28, 25] instead, works with triplets of images as its name suggests. It has similar applications to those of the contrastive loss and, in the last years, it has been widely adopted in person Re-ID. The triplets are called anchor, positive and negative samples. Intuitively, the anchor and the positive points are similar, while the anchor and negative points are different. There are several ways to choose the positive and negative samples. Hard mining relies on searching for the *hardest* positive and *hardest* negative to better understand how to recognize the same object [25]. This can be performed on the whole dataset, but it is computationally heavy and it could result in sampling too extreme samples or outliers. For person Re-ID, work [25] proposes an online hard batch sampling. By taking the hardest positive and negative samples in each batch, we are overall sampling *moderate* samples. Supposing that for each batch we randomly sample

C^* categories out of the total C classes and K images per category, we can write this loss as:

$$\mathcal{L}_{Tri} = \sum_{i=1}^{C^*} \sum_{k=1}^{K} \Big[m + \max_{p=1,\dots,K} D(f(\mathbf{x}_k^i), f(\mathbf{x}_p^i)) - \min_{\substack{j=1,\dots,P \\ n=1,\dots,K \\ i \neq j}} D(f(\mathbf{x}_k^i), f(\mathbf{x}_n^j)) \Big]_+ \tag{2.6}$$

where we used $f(\mathbf{x})$ instead of $f(\mathbf{x}, \boldsymbol{\theta})$ for brevity and D is the (squared) Euclidean distance of the embeddings. For each class i, for each anchor image k, the function takes the positive image with the same class having maximal distance from the anchor and the negative image over the other classes with the minimum distance from the anchor. In this way we compare, for each batch, the hardest positives and negatives relative to each anchor. The margin parameter m is also crucial in this setting, so as not to end up with trivial solutions. To summarize, this is equivalent to ask the difference between the negative and positive distances to be greater than the margin. The margin also prevents overly correcting already accurate triplets [25].

2.3.4 Performance metrics

As we just explained, the previous losses are functions of the parameters of a model. This means that we can exploit them to optimize the model itself, guiding the training process for the parameters updates. Besides them, there are other functions that are merely performance metrics, often giving different interpretations of how good a model is performing. One must pay attention on how to select these metrics. For instance, *accuracy* might give too optimistic results when a dataset is imbalanced. In chapter 3 we will explain the two most common metrics for pedestrian Re-ID. Below, we briefly show how accuracy works, common in classification. We can write it as:

$$\mathcal{A} = \frac{\#\text{ correct predictions}}{\#\text{ total predictions}} \tag{2.7}$$

where the ratio simply gives the fraction of times our model was right. Generally speaking, both performance metrics and loss functions are computed over a batch of data. However, they are usually reduced over a training iteration (*e.g.*, averaged or summed together), providing a more meaningful result that characterizes the entire epoch.

2.4 Optimization

Up until now, we saw how a neural network processes its input data. The samples flow from the first layer to the final one, in what is known as the *forward* pass.

Given a parameters configuration, this stage creates a feature representation of our inputs and eventually outputs a prediction. In this section, we will see how a neural network actually learns, updating its parameters. Starting from the last layer, we will update, step by step, the weights of each layer up until the first one. For this reason, this stage is known as *backward* pass.

2.4.1 Backpropagation

One can see a loss function as an objective function to minimize. In this sense, $\mathcal{L}(\mathbf{y}, f(\mathbf{x}, \boldsymbol{\theta}))$ is a function of some target $\mathbf{y}$ relative to the input $\mathbf{x}$ and of the neural network output given that input. Since the loss is a function of the model parameters, we can make use of this minimization to update the parameters themselves. This is done by exploiting the directional derivative $\mathbf{v}^T \nabla_{\boldsymbol{\theta}}(\mathcal{L}(\mathbf{y}, f(\mathbf{x}, \boldsymbol{\theta})))$ [18] (will will later see how to couple derivatives with respect to the input data and parameters). We want to find the best direction in which $\mathcal{L}$ decreases. Since the gradient points uphill, $\mathbf{v}$ will be the unit vector that points in the opposite direction of the gradient. We can translate this into the following iterative updating procedure, known as *gradient descent*:

$$\boldsymbol{\theta}' = \boldsymbol{\theta} - \epsilon \nabla_{\boldsymbol{\theta}} \mathcal{L}(\mathbf{y}, f(\mathbf{x}, \boldsymbol{\theta})) \tag{2.8}$$

where $\boldsymbol{\theta}'$ is the updated parameter and ϵ is the learning rate, a hyperparameter that indicates the size of the step. Intuitively, each time we update the parameters, we take a step in the direction that minimizes the loss $\mathcal{L}$ scaled by ϵ.

In a neural network, one would like to compute the gradients with respect to the parameters of the layers. However, it is not possible to directly solve this. To address this issue, we have to use the *backpropagation* algorithm that employs the chain rule as follows. During the forward pass, we track, for each layer, the gradients of its output relative to both its input and parameters. Then, in the backward pass, starting from the last layer (where we compute the loss) and ending in the first one, for each of them, we compute the derivative of the loss function with respect to both its inputs and parameters with the chain rule. Simplifying, if x is the input of a layer and o is its output, $\frac{\partial \mathcal{L}}{\partial x}$ can be decomposed into $\frac{\partial \mathcal{L}}{\partial o}\frac{\partial o}{\partial x}$. The term $\frac{\partial o}{\partial x}$ was computed during the forward pass while $\frac{\partial \mathcal{L}}{\partial o}$ is equivalent to the derivative of the loss relative to the input of the next deeper layer. In this way, we just need to multiply these terms and apply the same reasoning until we reach the first layer. Figure 2.6 provides a more complete example.

2.4.2 Regularization

As we mentioned before, it is crucial that a model should perform well on the test data distribution, without overfitting solely to the training one. In this sense, one can accept to introduce methodologies that might result in a higher training error

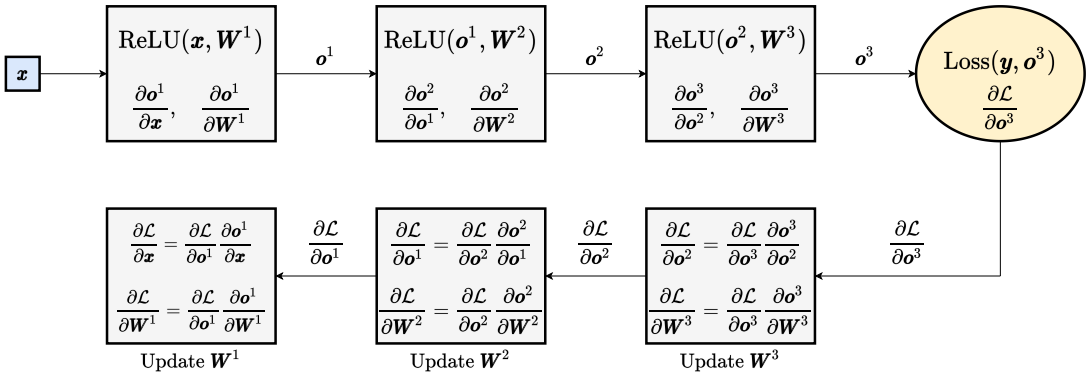

Figure 2.6: Backpropagation algorithm for a simple fully connected neural network. The input sample is $\mathbf{x}$ with some target $\mathbf{y}$ while the output of the i^{th} layer is $\mathbf{o}^i$ and $\mathcal{L}$ is a loss function. At each layer we exploit the derivative computed for the previous one (the immediately deeper layer).

but might reduce the test one. Among others, adding norm penalties of the model parameters to the loss function is common in this field. We might not include every single parameter belonging to $\boldsymbol{\theta}$ in the penalty [18]. We denote $\mathbf{w}$ as the vector of the model parameters that we want to regularize. The L2 norm penalty adds to the original loss function the term $\frac{1}{2}\|\mathbf{w}\|_2^2$ (where $\frac{1}{2}$ is useful for differentiation) and can be controlled with the hyperparameter λ. For a generic loss function $\mathcal{L}$, we would have $\tilde{\mathcal{L}}(\mathbf{y}, \{(\mathbf{x}, \boldsymbol{\theta})) = \mathcal{L}(\mathbf{y}, f(\mathbf{x}, \boldsymbol{\theta})) + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w}$. Consequently, eq. 2.8 becomes:

$$\mathbf{w}' = \boldsymbol{\theta} - \epsilon(\lambda\mathbf{w} + \nabla_{\mathbf{w}}\mathcal{L}(\mathbf{y}, f(\mathbf{x}, \mathbf{w}))) \tag{2.9}$$

where we just highlighted the update for the parameters that we want to regularize ($\mathbf{w}$). This is known also as *weight decay*, since it has the effect of penalizing large model weights (parameters), by decaying the weights themselves and thus constraining the training process.

2.4.3 Stochastic gradient descent

Stochastic gradient descent (SGD) is an optimization algorithm analogous to the standard gradient descent, with the difference that the gradient is not computed on the entire dataset, but rather on subsets of it. We already mentioned the concept of *batch of data*, meaning a small subset of our samples. Although there are different terminologies (*e.g.*, minibatch), in this work, we refer to a batch as a small subset of the training or test set with at least one sample, conventionally having an amount of data in the powers of 2. During each training iteration, *i.e* epoch, the dataset is divided into multiple (non-overlapping) subsets of size m. This is done

for technological limitations (memory in GPUs) but, possibly because of the noise that comes with small batches, it also brings better generalization capabilities, regularizing the training [18]. For each of those batches, when the examples are drawn randomly (*i.i.d.*), one can show that computing the average gradient over those m samples will produce an unbiased estimator of the true gradient (at least in the first epoch) [18]. For a given batch of data, we can write the gradient estimate as:

$$\hat{\mathbf{g}} = \nabla_{\boldsymbol{\theta}} \left(\frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(y_i, f(\mathbf{x}_i, \boldsymbol{\theta})) \right) \tag{2.10}$$

where the hat symbol highlights that this is an estimator of the true gradient. We then update the parameters similarly to eq. 2.8. After we optimized over all the minibatches, we need to repeat the whole procedure for a certain number of epochs. To speed-up the convergence, one can couple SGD with *momentum*. Momentum introduces the concept of velocity $\mathbf{v}$ which is equal to the exponentially decaying average of previous gradients. As explained in [18], there is a parallelism with the analogous physical concept. The contribution of previous gradients will force the optimization process to continue moving in their direction. Their exponential decay is a hyperparameter. The update rule requires now two steps:

$$\begin{aligned} \mathbf{v} &= \alpha \mathbf{v} - \epsilon \nabla_{\boldsymbol{\theta}} \left(\frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(y_i, f(\mathbf{x}_i, \boldsymbol{\theta})) \right) \\ \boldsymbol{\theta}' &= \boldsymbol{\theta} + \mathbf{v} \end{aligned} \tag{2.11}$$

where $\alpha \in [0, 1)$ is the momentum exponential decay hyperparameter, ϵ is the learning rate and $\mathbf{v}$ the velocity (which is initialized before the learning process). The current direction is influenced by the previous gradients, whose contribution depends on how large is α relative to ϵ [18]. A widely used alternative is *Neterov momentum*, where we first apply the current velocity and only then compute the gradient. Work [18] interprets this as adding a correction factor. We provide the algorithm of Nesterov momentum 2.2 since it summarizes this subsection well.

2.4.4 Adam optimizer

Another popular optimization algorithm is *Adam* [29]. This method combines the improvements of other optimization methods that extend the standard family of SGD algorithms. It scales the learning rate for each model parameter to have a faster or slower decrease depending on their partial derivative [18]. This learning rate adaptation depends on the exponentially decayed averages of both the first and second gradient moments (similar to the concept of momentum). The decay is controlled by the hyperparameters $\beta_1 \in [0, 1)$ and $\beta_2 \in [0, 1)$. Additionally, since the moving averages are initialized to zero vectors (and thus biased towards it),

Algorithm 2.2 Stochastic gradient descent optimizer with Nesterov momentum.

procedure SGD($\boldsymbol{X}, \boldsymbol{Y}, \mathbf{w}, f_{\boldsymbol{\theta}}, \epsilon, \alpha$)
  ▷ $\mathbf{x} \in \boldsymbol{X}$ are the datapoints while $y \in \boldsymbol{Y}$ the labels
  ▷ $(\mathbf{x}_i, y_i)$ is a tuple with a datapoint and its label
  ▷ The training dataset $\boldsymbol{X}$ is divided in m batches
  ▷ $f_{\boldsymbol{\theta}}$ is a neural network parameterized by $\boldsymbol{\theta}$
  ▷ ϵ is the learning rate
  ▷ $\alpha \in [0, 1)$ is the momentum exponential decay
  ▷ Initialize $\mathbf{v}$ and $\boldsymbol{\theta}$
  ▷ Execution
  while Not sampled all training batches **do**
    Sample batch $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ..., (\mathbf{x}_m, y_m)\}$
    $\boldsymbol{\theta}' \leftarrow \boldsymbol{\theta} + \alpha\mathbf{v}$ ▷ Apply current velocity before gradient
    $\mathbf{g} \leftarrow \frac{1}{m}\nabla_{\boldsymbol{\theta}'} \sum_{i=1}^{m} \mathcal{L}(y_i, f(\mathbf{x}_i, \boldsymbol{\theta}'))$ ▷ Compute gradient estimate
    $\mathbf{v} \leftarrow \alpha\mathbf{v} - \epsilon\mathbf{g}$ ▷ Update velocity
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \mathbf{v}$ ▷ Update parameters
  end while
  return $\boldsymbol{\theta}$ ▷ Return the optimized network parameters
end procedure

they counter their bias by correcting the moment estimates [29]. Algorithm 2.3 shows this procedure.

2.4.5 Transfer learning

Different from classical machine learning theory, deep neural networks are nonconvex functions. Convexity is a necessary condition not to incur in local minima when using gradient-based methods. However, even without the theoretical guarantees for learnability, when using larger datasets we can achieve great results. For smaller datasets, one could use *transfer learning* techniques [30]. In such a setting, we would like to exploit the weights learned on a different dataset or even task. For classification, it is common to use pre-trained neural networks on large datasets and *fine-tuning* on a smaller one. During the fine-tuning, we can freeze all the layers but the last one, considering them just as feature extractors. We can then extract the features from our samples and train the last layer from scratch for our task. Depending on the data availability and similarities with the original task, we can freeze more or fewer layers, considering the pre-training as a sort of parameter initialization. More advanced options include assigning lower learning rates to shallower layers (which capture more generic features) and more aggressive learning rates on deeper layers (more task-specific). All of this is neccessary to adapt the

Algorithm 2.3 Adam optimizer.

procedure ADAM($\boldsymbol{X}, \boldsymbol{Y}, \mathbf{w}, f_{\boldsymbol{\theta}}, \epsilon, \beta_1, \beta_2$)
  ▷ $\mathbf{x} \in \boldsymbol{X}$ are the datapoints while $y \in \boldsymbol{Y}$ the labels
  ▷ $(\mathbf{x}_i, y_i)$ is a tuple with a datapoint and its label
  ▷ The training dataset $\boldsymbol{X}$ is divided in m batches
  ▷ $f_{\boldsymbol{\theta}}$ is a neural network parameterized by $\boldsymbol{\theta}$
  ▷ ϵ is the learning rate
  ▷ $\beta_1, \beta_2 \in [0, 1)$ are the exponential decays for the two moment estimates
  ▷ Initialization
  $t \leftarrow 0$ ▷ Iteration counter set to 0
  $\delta \leftarrow 10^{-8}$ ▷ Small constant for numerical stability
  $\mathbf{m} \leftarrow \mathbf{0}, \mathbf{v} \leftarrow \mathbf{0}$ ▷ Initialize first and second moment estimators
  ▷ Initialize $\boldsymbol{\theta}$
  ▷ Execution
  while Not sampled all training batches **do**
    Sample batch $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ..., (\mathbf{x}_m, y_m)\}$
    $t \leftarrow t + 1$ ▷ Update iteration counter
    $\mathbf{g} \leftarrow \frac{1}{m} \nabla_{\boldsymbol{\theta}} \sum_{i=1}^{m} \mathcal{L}(y_i, f(\mathbf{x}_i, \boldsymbol{\theta}))$ ▷ Compute gradient estimate
    $\mathbf{m} \leftarrow \beta_1 \mathbf{m} + (1 - \beta_1)\mathbf{g}$ ▷ Compute first moment estimate
    $\mathbf{v} \leftarrow \beta_2 \mathbf{v} + (1 - \beta_2)\mathbf{g} \odot \mathbf{g}$ ▷ Compute second moment estimate
    $\mathbf{m}' \leftarrow \frac{\mathbf{m}}{1 - \beta_1^t}$ ▷ Correct bias of first moment estimate
    $\mathbf{v}' \leftarrow \frac{\mathbf{v}}{1 - \beta_2^t}$ ▷ Correct bias of second moment estimate
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \epsilon \frac{\mathbf{m}'}{\sqrt{\mathbf{v}'} + \delta}$ ▷ Update parameters (element-wise operations)
  end while
  return $\boldsymbol{\theta}$ ▷ Return the optimized network parameters
end procedure

network to our new data or task without forgetting what it has learned during the pre-training.

2.5 Architectures

In this section we briefly show some of the popular CNN architectures. It is important to understand how to combine the layers we explained to obtain a powerful function approximator. Through the recent deep learning history, CNNs sought more complex, deeper models, combining different operations. Building deeper models can bring overfitting and learnability issues. Until the adoption of residual connections, neural networks had to limit their number of layers.

2.5.1 Standard architectures

In this category, we find CNNs that stack the layers explained in section 2.2. The simplest architecture is that of the *LeNet5* [31]. It has two blocks made of convolutional and pooling layers, two fully connected layers and the final classifier. It employs the sigmoid activation function. A similar structure is that of *AlexNet* [32], famous for being the first CNN winner of the ImageNet [33] competition against shallow methods. It is deeper than LeNet5 (it has 5 blocks of convolutional and pooling layers) and adopts the ReLU activation. It uses layer normalization (modern implementations employ batch normalization) and dropout for the fully connected layers. A more interesting architecture is that of VGG [34]. These are deeper networks with either 16 or 19 layers that use ReLU nonlinearities, normalization and smaller 3×3 filters. Having such filters allows for the same receptive field size to have deeper networks with fewer parameters and more nonlinearities. This has set the general trend for deeper networks that employ smaller filters. Figure 2.7 shows the schematics of these networks.

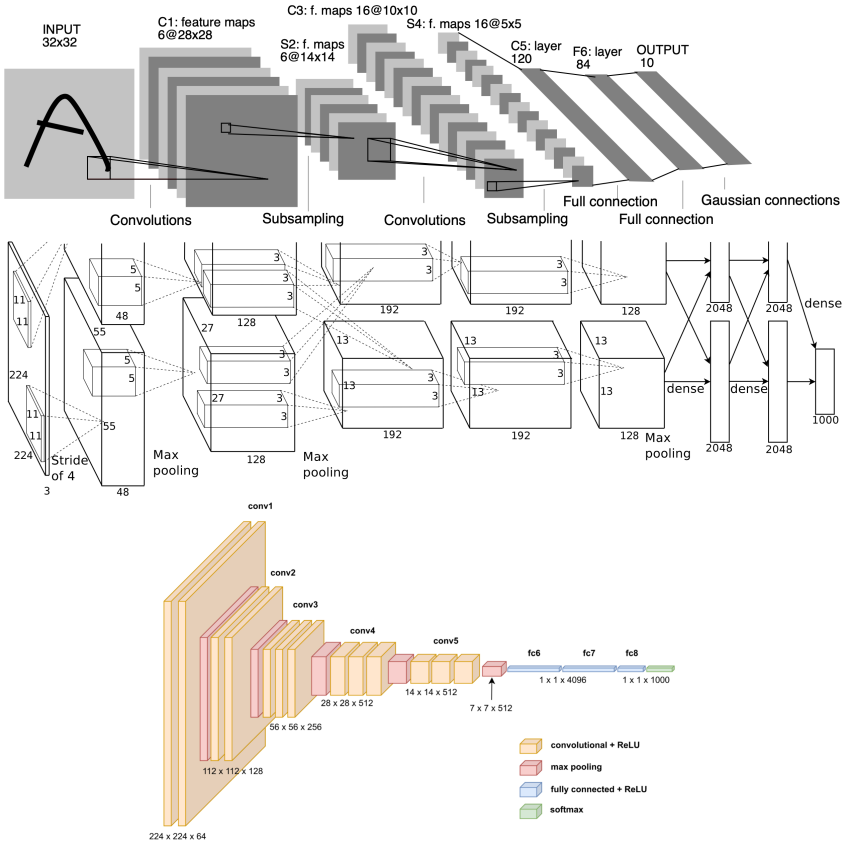

Figure 2.7: Overview of standard CNNs. First row: LeNet5 [31] architecture. Second row: AlexNet [32] architecture. Third row: VGG16 [34] architecture[1].

2.5.2 Residual connections

Work [4] shows how when just stacking layers to build very much deeper networks brings the degradation problem. This means that, beyond a certain point, adding layers starts worsening the performance metrics. The reason behind this is not overfitting (since, in that scenario, we would have a low error on the training data), but rather it is a symptom of the difficulty to optimize such models. For instance, suppose that we want to add layers to a less deep network and achieve the same score. The additional layers should learn the identity mapping. However, this is harder than it seems and the deeper model ends up with worsening solutions. The *ResNet* architecture [4] addresses this issue by introducing residual connections (see figure 2.8). For each block of convolution and pooling layers, they add the input to the output, $H(\mathbf{x}, \boldsymbol{\theta}) = f(\mathbf{x}, \boldsymbol{\theta}) + \mathbf{x}$, making it far easier to learn the identity mapping, *i.e.* $f(\mathbf{x}, \boldsymbol{\theta}) = 0$. In a conventional setting, these networks learn what is called the residual, $f(\mathbf{x}, \boldsymbol{\theta}) = H(\mathbf{x}, \boldsymbol{\theta}) - \mathbf{x}$. The shortcut connections can be seen as highways where the gradient can flow, mitigating exploding and vanishing gradients. When more than 50 layers are needed, ResNets employ the bottleneck structure to reduce the number of parameters.

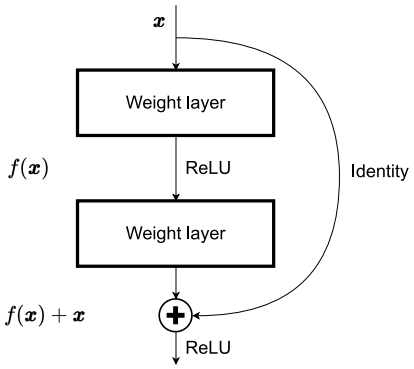

Figure 2.8: Residual module of ResNet, adapted from [4].

2.5.3 Other architectures

There have been countless research on model architectures, delivering several neural network arrangements. We have recently seen advances on attention modules and non-local operators [35, 36, 37], temporal dependency modeling [38, 37] and efficient neural networks [39]. Another interesting architecture is *GoogLeNet*, based on

[1]The VGG16 image is from:
`https://github.com/kennethleungty/Neural-Network-Architecture-Diagrams`

the *inception* module [40, 41]. This family of networks has a *network in network* structure [20], stacking multiple modules that can be seen as smaller local neural networks. These modules have parallel convolutional layers (with also nonlinearities and pooling layers) that allow different receptive fields. In figure 2.9, we illustrate the inception module with bottlenecks (reducing the channel size).

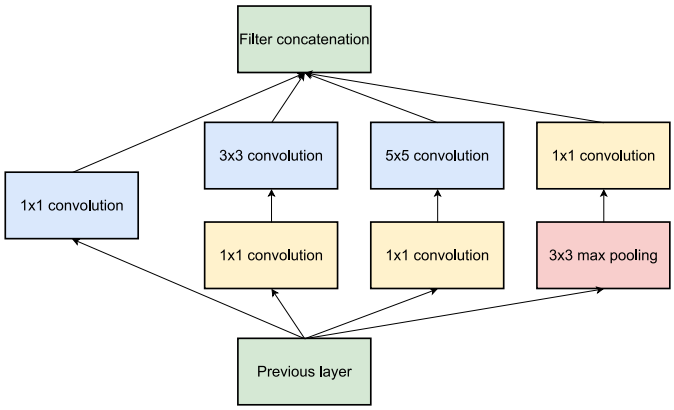

Figure 2.9: Inception module of GoogLeNet, adapted from [40].

2.6 Generative adversarial neural networks

Up until now, what we explained falls under the paradigm of discriminative learning. In a standard classification scenario, what we care about are the predictions, *i.e.*, estimating the conditional probability of the labels having observed the data. However, one might be interested in generating new data. In this case, we would like to estimate the underlying data distribution or the joint probability distribution. Explicitly estimating the density would easily allow us to sample new data points. Unfortunately, this is not always feasible, but there are several paradigms for generative learning, as shown in figure 2.10. In this work, we will only focus on *generative adversarial neural networks* (GANs). In their plain form, we sample from a simple distribution, generally random noise, and learn the transformation to the desired outputs (the images). The transformation itself is a neural network (although there are other approaches) trained so that the outputs are aligned to the already available (training) data distribution.

2.6.1 A two-player game

First introduced in [43], GANs have gained popularity in several computer vision fields for their capabilities and expressiveness. In this work we will only refer to deep convolutional GANs [44], although there exist also other frameworks. To

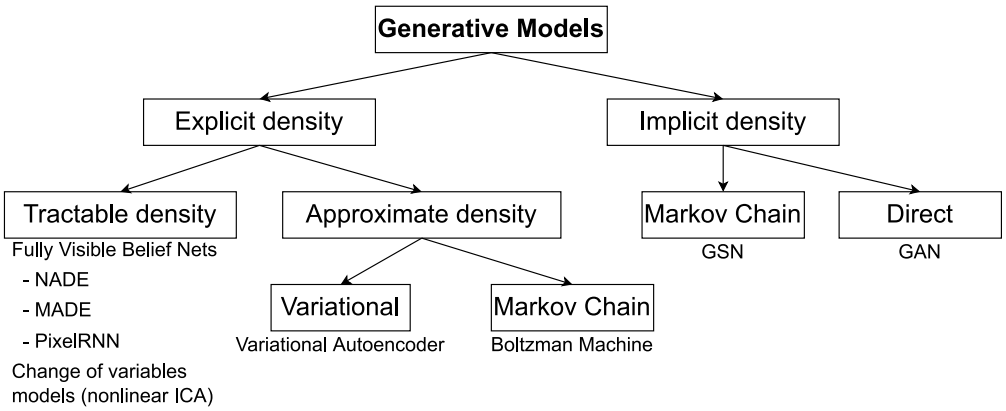

Figure 2.10: Taxonomy of generative models, adapted from [42].

better understand their inner workings one can think of them as a two-player game. The two *adversaries* are commonly known as generator G and discriminator D. The latter operates like a traditional neural network. Its objective is to predict whether an image is real or fake, *i.e.*, if it belongs to the available data or if it was generated. On the other hand, the generator tries to *fool* the discriminator by counterfeiting images to be indistinguishable from the real ones. It operates by sampling (Gaussian) random noise $\mathbf{z}$ (latent variable) and outputting an RBG image (see image 2.11 for an example).

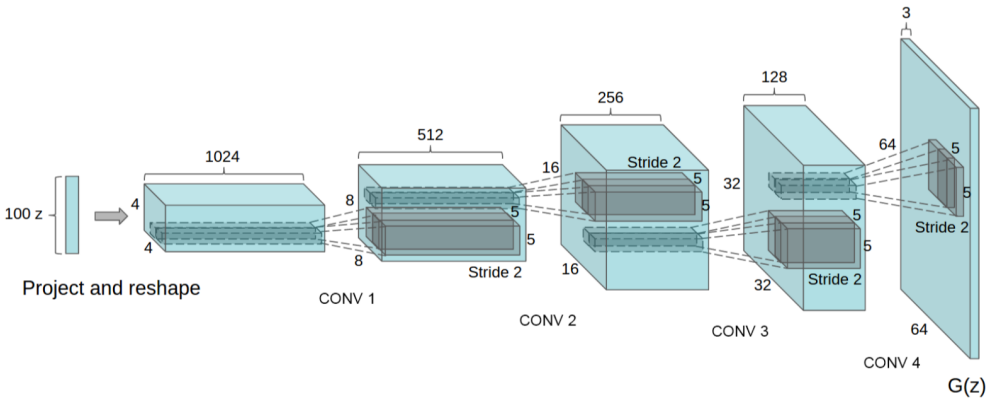

Figure 2.11: Generator structure for deep convolutional GANs [44]. It samples a 100-dimensional (uniform) random vector and then maps it into a 64×64 pixel image through several transposed convolution layers.

2.6.2 Adversarial training

Considering a generator G parameterized by $\boldsymbol{\theta}^G$ and a discriminator D parameterized by $\boldsymbol{\theta}^D$, as noted in [43] we can frame the problem as a min-max game with value function $V(G, D)$:

$$\min_G \max_D V(G, D) = \mathbb{E}_{\mathbf{x}\sim P_{data}}\Big[\log D(\mathbf{x})\Big] + \mathbb{E}_{\mathbf{z}\sim P_{\mathbf{z}}}\Big[\log(1 - D(G(\mathbf{z})))\Big] \tag{2.12}$$

where each input image $\mathbf{x}$ is sampled from the empirical data distribution P_{data} and the latent variable $\mathbf{z}$ is sampled from the latent space distribution $P_{\mathbf{z}}$. The output of $D(\mathbf{x})$ is the probability ($\in$ [0,1]) that the input image is real (with label 1) or fake (with label 0) while the output of $G(\mathbf{z})$ is a generated image given $\mathbf{z}$. This means that, on average, from the discriminator's point of view, we would like to maximize the probability of correctly distinguishing fakes from real images. From the generator's point of view instead, we would like to minimize the discriminator's probability of predicting as fake the generated images. During each training iteration, there are two batches of data available, the real training images, and the generated ones. Since there are two networks, we have to perform two gradient updates per iteration (discriminator and generator can be optimized by different algorithms, with different hyperparameters such as learning rate and weight decay). However, we point out that alternative approaches update the two networks at different steps. For the discriminator objective in eq. 2.12, the weighted binary cross-entropy loss [42] allows standard gradient descent:

$$L^D(\boldsymbol{\theta}^{\boldsymbol{G}}, \boldsymbol{\theta}^{\boldsymbol{D}}) = -w_r\mathbb{E}_{\mathbf{x}\sim P_{data}}\Big[\log D(\mathbf{x})\Big] - w_f\mathbb{E}_{\mathbf{z}\sim P_{\mathbf{z}}}\Big[\log(1 - D(G(\mathbf{z})))\Big] \tag{2.13}$$

where w_r and w_f are the weights relative to the discriminator's ability to correctly assign the label 1 when fed real images and 0 otherwise. Usually $w_r = w_f = \frac{1}{2}$. When minimizing this expression from the discriminator's point of view, the generator's weights remain fixed, and vice versa. For the generator loss instead, there are more caveats. One could apply gradient ascent (thus maximizing) on the second term of eq. 2.13 to fool the discriminator, or equivalently minimize the likelihood as in eq. 2.12. However, as noted in [42, 43], optimizing the expression $log(1 - D(G(\mathbf{z}))$ from the generator's perspective would produce weak gradient signals when the generator is doing a poor job (fake images are easily classified as so) and a strong gradient signal when the generator is outputting indistinguishable images. With these remarks, we can flip the target labels and construct a binary cross-entropy loss [42] for the generator:

$$L^G(\boldsymbol{\theta}^{\boldsymbol{G}}, \boldsymbol{\theta}^{\boldsymbol{D}}) = -w_r\mathbb{E}_{\mathbf{z}\sim P_{\mathbf{z}}}\Big[\log D(G(\mathbf{z}))\Big] \tag{2.14}$$

where usually $w_r = \frac{1}{2}$. This might seem confusing, but all we did was switch to the generator's perspective where the discriminator should output 1 (the real label

from this point of view) when it is fed a fake image. We can now apply standard gradient descent techniques and iterate the whole procedure.

When training a conventional neural network, one is actually solving an optimization problem, which hopefully will converge to a local minimum. The adversarial perspective we just explained, is theoretically framed as a game, whose solution is a Nash equilibrium. As described in [42], this equilibrium is a local minimum of L^D with respect to $\boldsymbol{\theta^D}$ and a local minimum of L^G with respect to $\boldsymbol{\theta^G}$. If we define as P_G the generator distribution, *i.e.*, the estimated data distribution implicitly defined by G with respect to the original one (P_{data}) when $\mathbf{z}$ is sampled from $P_{\mathbf{z}}$, such equilibrium occurs when $P_G = P_{data}$. The authors of [43] state that, when using a model of infinite capacity (non-parametric setting), the min-max game converges to a global optimum when the previous equality holds. Similarly to what we explained above, they first fix the generator and find the optimal discriminator. Then, P_G is updated to optimize $V(G, D^*_G)$ of eq. 2.12, where D^*_G is the optimal discriminator given G. They prove that, with this procedure, P_G will converge to P_{data}. However, in practice one will employ CNNs and MLPs for the discriminator and generator, coupled with gradient-based methods for the optimization. Since we are refusing the initial assumptions (critical points, convexity, capacity), this will violate the theoretical guarantees [43]. Having said that, these architectures are pervasive in GANs and have shown, during their evolution, excellent performance.

2.6.3 Beyond binary classification

Since their introduction, GANs gained momentum and popularity, resulting in advances in both theoretical and applied research. These networks have since been designed with different losses (*e.g.*, Wasserstein GAN [45]), for several tasks such as super-resolution [46] and image translation [47, 7, 6]. As a natural extension of the binary cross-entropy loss, work [42] presents (one-sided) label smoothing. As reported by the authors, this should counter extreme confident predictions, encouraging soft probability estimates from the discriminator. Similarly, one could employ the least-square (mean square error) loss. Work [48] encourages this function essentially for two reasons. Contrary to the cross-entropy loss, the least-square loss will continue to push, when optimizing the generator, the fake samples closer to the decision boundary, even when they already are on the correct side. This is especially useful for those samples that are correctly classified and lie far away from the decision boundary. Additionally, by doing so we are increasing the gradient signal when optimizing the generator, relieving the vanishing gradient problem which can occur when training with cross-entropy [48]. The generator and discriminator

losses can now be expressed as:

$$L^D(\boldsymbol{\theta^G}, \boldsymbol{\theta^D}) = \frac{1}{2}\mathbb{E}_{\mathbf{x}\sim P_{data}}\Big[(D(\mathbf{x})-1)^2\Big] + \frac{1}{2}\mathbb{E}_{\mathbf{z}\sim P_{\mathbf{z}}}\Big[(D(G(\mathbf{z})))^2\Big] \tag{2.15}$$

$$L^G(\boldsymbol{\theta^G}, \boldsymbol{\theta^D}) = \frac{1}{2}\mathbb{E}_{\mathbf{z}\sim P_{\mathbf{z}}}\Big[(D(G(\mathbf{z}))-1)^2\Big] \tag{2.16}$$

where we used the label 1 for real images (note that in the second equation we are in the generator's perspective, we want the discriminator to believe that the generated images are real) and 0 for the fake ones. As it will be shown in chapter 5, we adopted this loss in our framework. Before moving on the next chapter, we conclude this one with a series of suggestive images (figure 2.12) generated with GANs. For GANs evaluation methods, see appendix B.

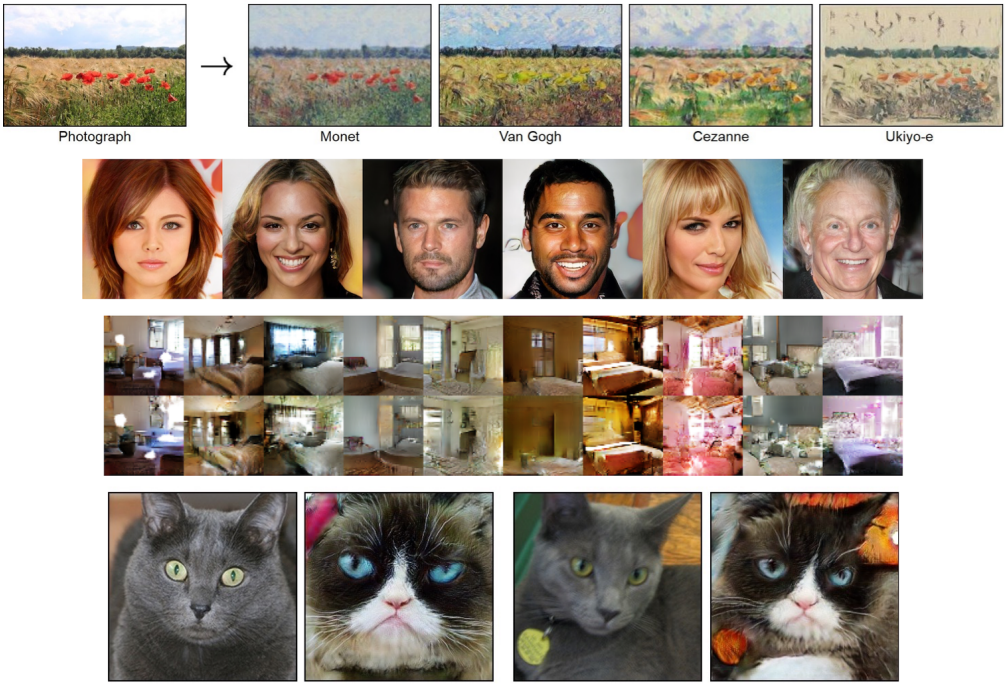

Figure 2.12: Images generated with GANs. First row: style transfer from photo to painting [7]. Second row: realistic human faces generated by [49]. Third row: furniture generation [44]. Fourth row: style transfer from blue to grumpy cat [6].

Chapter 3

Pedestrian Re-Identification

In the previous chapter, we reviewed the main building blocks and architectures of modern neural networks. We can now study how such models were developed and optimized for the task of pedestrian re-identification (Re-ID). This chapter will start by stating what we are trying to achieve with this kind of task and how it is framed by means of *classification* and *image retrieval*. We then provide a brief evolution of techniques that address this problem, ranging from handcrafted image descriptors to deep convolutional neural networks. In particular, the latter will be analyzed more deeply, since they represent the state of the art.

This task has gained momentum since it has practical applications in high risk areas where security is needed. At the same time, it has also raised ethical concerns, because of its sensitive connotations, leading to important paradigm shifts such as adopting synthetic datasets. Since the pedestrians do not share the same identity between training and test set of the same dataset, is often not enough to use standard baselines, due to the large variations between datasets. This of course amplifies when one performs cross-domain analysis (as we will explain in the next chapter).

3.1 Person Retrieval

Person Re-ID aims at identifying a person of interest across different camera views [50, 51]. It can be thought of as a particular case of *content-based image retrieval* (CBIR) where, instead of having several objects, we need to match different pedestrian identities. As for CBIR, given a query, the retrieval process consists in running it against a database called gallery and ranking the results by similarity. The query can be a single image or multiple stacked frames that depict an individual from a certain camera view. To make the task more interesting, the query is usually matched against the gallery without considering the frames of that person extracted

from the query camera view. Alternatively, if it is not possible to capture the same individual at least across two camera views, we can match the query against a gallery that contains images of the person of interest taken at a different time instant than that of the query [50]. Matching the correct person can unravel many challenges since a specific pedestrian is likely to be depicted under many viewpoints, with different lighting conditions, pose, imperfect bounding boxes and occlusions across the frames. Occlusions do not happen only with objects, we could have body parts of other people in the same image. On top of this, there could be other pedestrians in the background, rendering the whole task even more challenging.
Differently from the general setting of CBIR, datasets for person Re-ID provide a labelled training set. This means that we can train a model for classification on that set. The trained model will then be used as a feature extractor on the query and gallery sets for the retrieval process. Identities for the classification task are different than those employed during the retrieval. One must be careful enough not to overfit only on the training set, even if the ambient conditions and other data characteristics are similar among the different splits.

3.1.1 Building blocks

It is now reasonable to explore the main building blocks of a traditional person Re-ID system (shown in figure 3.1). We can consider four main components, where each of them could represent a problem of its own. Some of these modules will be detailed in later sections. As of right now, we can draft an overview.

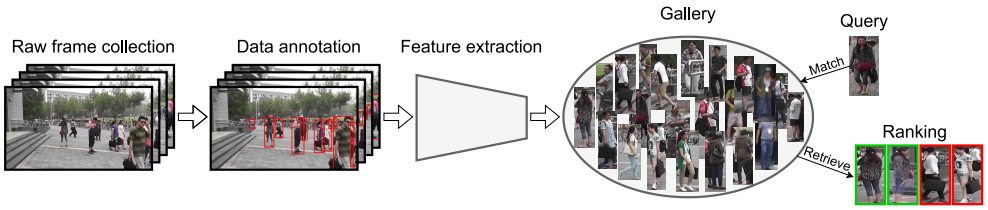

Figure 3.1: Standard person Re-ID pipeline. The steps are as follows:
1) raw dataset collection, 2) data annotation, 3) feature extraction, 4) pedestrian retrieval. The image frames were taken from the PRW dataset [52].

1. The first step concerns the raw video data collection. In a real environment, we would like to capture the same individual across at least two viewpoints. Cameras therefore must be placed strategically.

2. After having collected all the necessary raw data, one of the most crucial aspect is the annotation of each frame. We need to extract the bounding boxes of each pedestrian and assign them an identity. The first step can be either

done by hand such as in [53] or with object detection techniques as in [1] or [3]. The second one instead lies in manually matching the so found bounding boxes of pedestrians with the same identity. This process is time consuming and error prone. If not done carefully, it could undermine the quality of the entire dataset.

3. When all the bounding boxes have been generated, it is possible to proceed with the feature extraction. Even if this started by handcrafting features to describe the pedestrians, as of today, the state of the art is dominated by CNNs. In the next section there will be a brief evolution of the techniques that have been used to tackle this task.

4. The last step is known as *pedestrian retrieval*. After having extracted the feature representations of both the query and gallery image, it is finally possible to match the embedding of the former to that of the latter by some similarity measure and obtain a ranking.

One might be tempted to create a system which solves all of the aforementioned problems (*end-to-end* person Re-ID). In reality, those are treated as different tasks. For instance, the second step is known as *pedestrian detection* [54]. To make large scale datasets possible and reduce annotation time, bounding box generation is often performed with human detectors such as deformable part models (DPM) [55] or other *object detection* pipelines, rather than by hand. The resulting bounding boxes will not be perfect (some body parts can be cropped-out or the region includes too much background, adding challenges to the retrieval task) and, therefore, some supervision is still needed to choose the best ones. As we can see, this pipeline is not devoid of issues and it is reasonable to consider it as a separate task.
When reading the related literature, one will find that recent works mainly focus on pedestrian retrieval, including the extraction of strong feature descriptors of the original images. Consequently, this work will focus on those two last building blocks and, from now on, unless stated otherwise, pedestrian Re-ID will indicate only that part of the entire pipeline.

3.1.2 Open versus closed world

As we just saw, there are many approaches to pedestrian Re-ID. Work [50] provides a practical distinction between *closed* and *open* world Re-ID (table 3.1). In an open-world scenario, one could try to embed different kinds of data, such as depth maps, consider unmatched queries or include the human detection task (end-to-end person Re-ID), starting from the raw frames. Other relevant challenges comprehend noisy annotations (which is not uncommon on a small fraction of the data, even

Closed-world	Open-world
Single-modality Data	Heterogeneous Data
Bounding Boxes Generation	Raw Images/Videos
Sufficient Annotated Data	Unavailable/Limited Labels
Correct Annotation	Noisy Annotation
Query Exists in Gallery	Open-set

Table 3.1: Possible differences between open and closed world person Re-ID. This table was borrowed from [50].

unintentionally) and unavailable labelled data (crucial aspect when we want to bridge the gap between different domains).

The most recurring setting of person Re-ID in the literature is finding a model that works well on a specific dataset, exploiting the already extracted bounding boxes. Even though this could be thought of as a closed-world scenario, it is not always a given that the instances where correctly annotated. Additionally, increasingly more works try to bridge the gap between two or more datasets, addressing the problem of unavailable labelled data. This is extremely important in such a practical task, since it would not be ideal to train a different model based on the location of a security camera, for instance. Thus, one can argue that more recent works lie somewhere in the middle of these two extrema.

3.2 Training protocols and evaluation metrics

Modern person Re-ID systems adopt some kind of CNNs for the feature learning process. As we will see in the next section, models range from simple feed-forward neural networks to generative and complex architectures, which are optimized with several loss functions. Datasets are usually split into training and test sets with different identities. One can train a neural network in a supervised fashion on the training partition of identities. However, since the test set does not share the same identities, we cannot just treat the inference stage as a classification problem. The test set is further divided into query and gallery sets. We first feed the trained network with all the images in the gallery, extracting only a feature vector descriptor instead of their class probabilities. This can be done by excluding the classifier layer during inference, hoping that our model will deliver strong descriptors for unseen identities. Then, for each (probe) image in the query, we extract its feature vector and rank the gallery images from most to least similar. Euclidean distance and cosine similarity between the probe and gallery images are often adopted to generate such ranking. This is also known as *single-shot* since only one query image

at a time is used. Had we considered multiple queries at the same time, we would have been in a *multi-shot* scenario. To increase the performance, one could apply re-ranking as explained in [56, 57]. In the following subsections, we will explain and compare the two most adopted metrics to evaluate the ranking result (we only consider single-shot settings). Unfortunately, we do not often see a validation step in this pipeline or if it present, there is not a unified approach. A simple way of performing this would be to allocate a small number of images for each training pedestrian to the validation set, as in a standard classification pipeline. However, we argue that this would not be representative of the final query-gallery search task, also because we would validate on images of seen pedestrians. A more interesting approach could be to take out an appropriate number of identities from the training ones and construct a validation set with query and gallery images. Following this direction, we would validate on unseen identities, evaluating in the same fashion as in the test phase.

3.2.1 Cumulative matching characteristics

Having run a set of queries against a gallery set of images, one would like to know how good is such ranking. Specifically, we would prefer to have the gallery images with the same identity of the probe one at the beginning of the ranking. A simple way to evaluate how good our model is in this scenario relies on extending the notion of accuracy. Instead of being interested only in the top-1 accuracy, we could select a set K of rankings (usually $K = \{1, 5, 10, 20\}$) and, for each of $k \in K$, return the top-k accuracy. For a given query q, define the following step function:

$$f_k(q) = \begin{cases} 1 & \text{if query identity is in top-k ranking,} \\ 0 & \text{otherwise} \end{cases} \tag{3.1}$$

where $k \in K$. The CMC curve is then obtained by averaging the step functions of each query, at the considered ranks.

3.2.2 Mean average precision

Differently from CMC, the *Mean average Precision MaP* provides a single value (the higher, the better) which summarises the quality of the whole ranking. At least in this field, it is considered more reliable, however, at the same time, it is harder to obtain high values of MaP, rather than rank-1 accuracy. To determine its value, we have to compute, for a set of queries, the mean of the *average precision* scores for each query[1] by integrating its *precision-recall curve*. In practice, this is

[1]There are different interpretations of the MaP. We followed the standard integral approximation which has been extensively employed not only in person Re-ID, but also

replaced by a finite sum. To formalize, the recall can be defined as

$$recall = \frac{TP}{TP + FN} = \frac{\#\text{ correct relevant elements}}{\#\text{ relevant elements}} \tag{3.2}$$

where TP are the true positives and FN the false negatives. The precision instead can be defined as:

$$precision = \frac{TP}{TP + FP} = \frac{\#\text{ correct relevant elements}}{\#\text{ retrieved elements}} \tag{3.3}$$

where FP are the false positives (the rest as in eq. 3.2). Thus, for a given query, the average precision is computed as follows:

$$AP = \sum_{k=1}^{NRI} \Big(P(k)\Delta r(k)\Big) = \frac{\sum_{k=1}^{NRI} \Big(P(k)R(k)\Big)}{NRI} \tag{3.4}$$

where k is the rank (elements retrieved), NRI is the number of relevant elements in the ranking, $P(k)$ is the precision at rank k, $\Delta r(k)$ is the recall change between rank k and $k-1$, while $R(k)$ is 1 if the image at rank k is relevant, zero otherwise. Both formulations in eq. 3.4 return the same result. To compute the MaP, we can now average the previous equation over all queries, obtaining:

$$MaP = \frac{\sum_{q=1}^{Q} AP(q)}{Q} \tag{3.5}$$

where Q is the total number of queries.

3.3 Benchmark datasets

We can now review some of the datasets for pedestrian Re-ID. For each of them, we provide a brief description of their salient characteristics. Table 3.2 provides a summary of their main characteristics.

3.3.1 Market1501

First introduced in [1], it includes 1,501 identities depicted in 32,668 bounding boxes. The dataset was recorded by six (overlapping) cameras, five 1280×1080 HD

in the more general setting of image retrieval. For more details, one can refer to the following link: `https://en.wikipedia.org/wiki/Evaluation_measures_(information_retrieval)#Average_precision`.

cameras and one 720 × 576 SD camera, at a campus supermarket. Each individual has been captured by at least two camera views for cross-camera search.
Although they also extracted hand-drawn ground truths, the bounding boxes were generated with the DPM detector [55] as follows. For each candidate detected bounding box, they compute the *intersection over union* with the hand-cropped ground truth. If this ratio is higher than 50%, the detected box is marked as "good", if the ratio is smaller than 20% the bounding box is marked as "distractor", otherwise, it is marked as "junk" (*i.e.*, it will not influence process). "Distractors" also include identities that do not belong in the original 1,501 ones. They can be viewed as "false alarms" that provide additional challenges to the retrieval process (see figure 3.2). However, they are not always included in the Re-ID task. Nonetheless, as was mentioned earlier, having "imperfect" bounding boxes increases the challenges for the image retrieval, rendering it more representative of a real-world scenario. They also provide multiple query and gallery images for the same identity. From now on we will refer to as the Market dataset.

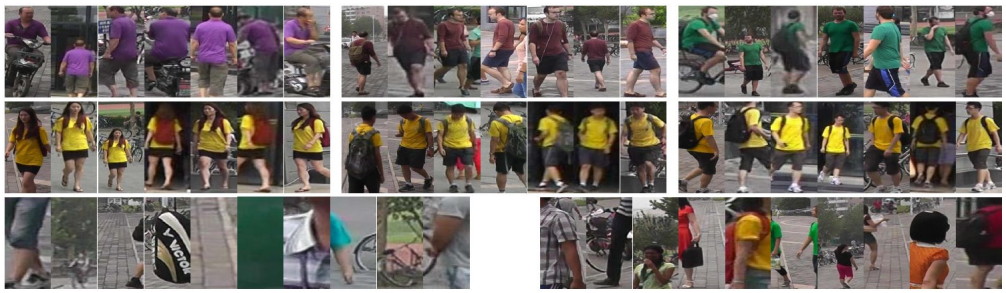

Figure 3.2: Market1501 dataset [1]. *First row:* three individuals with different appearance. *Second row:* three individual with similar clothing. *Third row:* examples of "distractors" on the left and "junk" images on the right.

3.3.2 DukeMTMC

This dataset was first introduced in work [58]. It serves many purposes since the authors extracted both semi-automatically generated bounding boxes and manually annotated path trajectories. It was recorded by eight 1080p cameras (85 minutes each) at the Duke University campus. They extracted 6,791 trajectories for 2,834 identities from views that are mostly disjoint, even if some overlappings are present. There are several occlusions and blind spots and pedestrians share different appearance.

For person Re-ID, in the literature subsets of the original DukeMTMC such as DukeMTMC-Reid [2] or DukeMTMC4ReID[59] are more common. In the former, we can find a total of 36,411 bounding boxes related to 1,812 identities, where 1,404

of them share at least two camera views while the remaining 408 ones that appear in one camera only can be used as distractors. They split in half the pedestrians for the training and testing set, for which they query one image for each identity for each camera against all the remaining gallery. In the latter instead, they consider 1,852 identities with 1,413 of them that appear in two or more cameras and 439 distractor identities which appear in only one camera view. They follow a procedure similar to that of the Market dataset for the bounding box annotation. Since the most common version is the DukeMTMC-Reid, from now on we will refer to it when we mention the Duke dataset.

3.3.3 CUHK03

This dataset [3] contains 14,097 images of 1,467 pedestrians that have been captured by a total of 6 cameras. Each pedestrian has been recorded by two different camera views, for an average of 4.8 images per camera. The bounding boxes were detected both manually and with pedestrian detectors, to increase realism (inaccurate detections). We will only focus on the version with detected bounding boxes. Features such as misalignment, occlusions and body part missing are recurrent, rendering the pedestrian retrieval more difficult. Since the videos were collected over months, illumination changes depend on multiple factors (such as weather) and are frequent even in the same camera view. Additionally, cameras have different settings and form complex cross-view transforms.

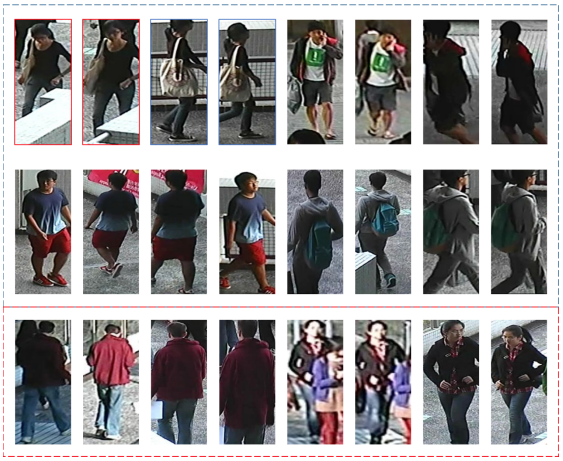

Figure 3.3: Image samples from CUHK03 dataset [3].

Dataset	Market	Duke	CUHK03
Bounding boxes	32668	36411	14097
Bounding boxes type	Detected	Hand-drawn	Detected
Total cameras	6	8	6
Total identities	1501	1812	1467
Training identities	750	702[2]	767
Testing identities	751	702[3]	700

Table 3.2: Summary of the main datasets characteristics.

3.4 From shallow to deep person descriptors

In this section, we will finally provide the highlights that, over time, made this ever-growing task as we know it today, starting from the techniques of the early days and shedding light on the most recent architectures. After doing so, we will unravel what constitutes the current *state of the art*. This chapter is focused only on methodologies limited to training and testing on the same dataset. This subtask is known as *supervised person Re-ID*. Cross dataset analysis will be the main argument of the next chapter. The training procedures extend to many recent learning paradigms and their performance evaluation is coupled to the benchmark datasets of choice. It cannot be taken for granted that a model will achieve a competitive performance on all of those datasets. Tables 3.3, 3.4 and 3.5 show how the models perform on the aforementioned datasets.

3.4.1 Architecture evolution

Early methods for person Re-ID relied on the manual extraction of pedestrian descriptors. Most works exploited color features [51], working with the weighted color histogram in different color spaces such as in [60]. Such strategies are aimed at differentiating the pixels belonging to the pedestrian from the background ones. Later works extracted more robust descriptors such as *sift* [14] or *surf* [15]. Despite all the progress in hand-crafted features, currently adopted methods rely on deep learning techniques. At first, only standard CNNs for classification were employed. However, over time contrastive methods gained popularity. Siamese [26] neural networks are helpful for contrastive losses since they work with pairs of images

[2]The remaining 408 identities over the 702 for training and 702 for testing are used as distractors.

[3]See footnote 2.

and have been heavily adopted in this field. Other important losses then appeared. Work [25] underlines the importance of the triplet loss for mining hard intra-batch samples. As of today, one will notice a combination of multiple criteria for neural network training, such as cross-entropy and triplet loss. In the next subsections, we will present the most recent works, showing the heterogeneity that exists in these methods.

3.4.2 Image cues

The most common approach that one can find in the literature lies in extracting visual features from the cropped bounding boxes. Recent works try to minimize both the identity loss (cross-entropy loss with the training identities) and other ones such as the triplet loss.

In work [61], the authors provide a *Bag of Tricks* (BoT) to extend the discriminative capabilities of an Imagenet pre-trained [33] ResNet [4]. Among others, they propose the "BNNeck" (shown in 3.4), which adds a batch normalization [22] layer after the global averaging pooling. They argue that, when optimizing simultaneously the identity and triplet losses, considering their inconsistent goals, when one is reduced the other one could keep oscillating or diverge. Normalizing the features before feeding them to the final classifier will constrain them on a hypersphere and thus, make it easier for the cross-entropy loss to converge. Depending on the features employed for the inference ranking (before or after the normalization), one should consider Euclidean distance or cosine similarity. They also experiment with random erasing, warmup learning rate and different last stride sizes.

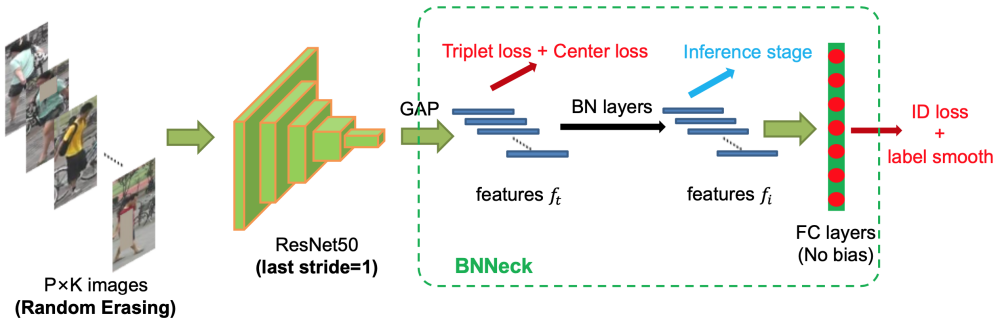

Figure 3.4: BNNeck structure [61]. P and K stand respectively for the number of identities and number of images per identity in each batch.

Part-based models decompose an image into multiple regions which represent different body parts. Work [62] proposes the architecture (PCB) shown in figure 3.5. The authors removed all the layers starting from the global average pooling of the backbone CNN (such as a ResNet) and divided the spatial dimensions of

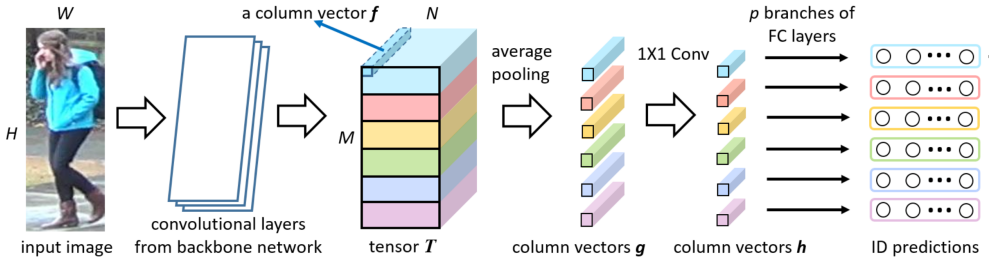

Figure 3.5: Structure of the standard PCB architecture [62].

the resulting features (with C channels) into s horizontal stripes. For each stripe, average pooling is performed, rendering p C-dimensional vectors. Before feeding each of such vectors to a separate identity classifier, they apply 1×1 convolutional filters to reduce the channel size. When dividing the spatial dimension into stripes, we are coercing all the column vectors (see 3.5) to be in the same region. Some of them might be more similar to the column vectors of other parts. To address this "within-part" inconsistency, the authors also propose a *refined part pooling* (RPP) that relocates by similarity those "outliers".

Other methods try to erase regions of the images forcing the network to learn discriminative features from different image regions or body parts. For instance, work [63], instead of removing random regions, drops the features with the largest activations, pushing the network to encode discriminative features from low informative regions. To overcome noisy results, they also introduce a regularizing stream and a global stream that operates on the original pictures.

One common issue that characterizes almost all Re-ID datasets is the distribution gap between the different cameras. To overcome this issue, work [64] theorizes a new module that can be easily embedded in different kinds of CNN architectures. They design a camera-based batch normalization layer (CBN) that exploits camera annotation to "disassemble" the datasets and align different camera distributions. During training, instead of normalizing with the batch statistics, they normalize separately the instances belonging to different cameras with their mean and standard deviation. During testing instead, they collect an unlabeled set of images and compute the statistics for each testing camera.

3.4.3 Generative methods

As we saw in the previous section, a certain individual could appear under different illuminations, poses and viewpoints. Up until this moment, the proposed methods tried to reduce the influence of intra-class variations by minimizing more losses, dividing the feature maps into multiple regions or more in general modifying existing backbone CNNs. With GANs instead, to increase the robustness of a model, one

could generate reasonably realistic images which are diverse enough to make up for unseen variations [65].

In work [66], the authors adopted a siamese [26] neural network, which they called FD-GAN (pose-guided feature distilling GAN), where each of the two branches includes an image encoder, an image generator, a pose encoder, an identity discriminator and a pose discriminator. As it is shown in figure 3.6, upon receiving two images of the same person, each branch will first extract the visual features and, subsequently, generate a new image conditioned on its identity and a target pose (provided as a vector embedding). The generator, together with the image encoder, would like to fool both the identity and pose discriminators, with the aim of learning identity-related and pose-unrelated visual features. To better regularize this process, the extracted feature from the two images are merged into a verification classifier (which checks if they belong to the same person), while a novel *same-pose* loss between the generated images is introduced.

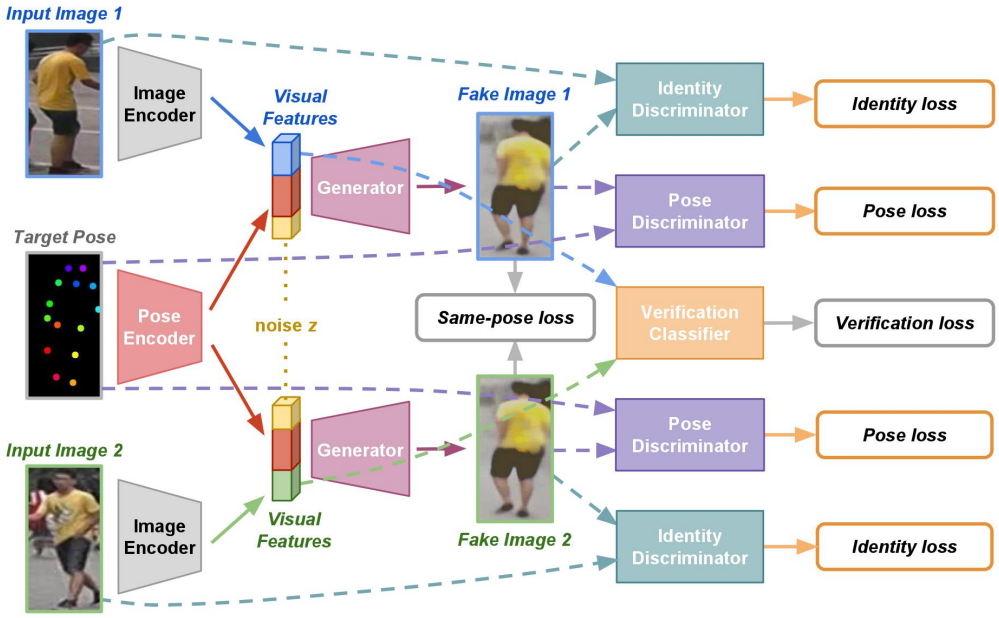

Figure 3.6: Overview of FD-GAN [66]. The target pose is represented by a vector embedding. We can see the many modules and task that characterize this network. The verification and same-pose loss have been used to better regularize the whole process.

A different joint discriminative and generative approach that does not require additional pose embeddings can be found in DG-Net [65]. They introduce a novel generative module, made up of an appearance and a structure encoder, a decoder (the generator) and the discriminator. The two encoders respectively decompose

each pedestrian in what they call the appearance and structure latent space (code). The former describes id-related cues such as clothing color, texture and style, while the latter describes the background, hair, pose, viewpoint and body size. Intuitively, after the latent codes have been extracted from two images, they feed the generator with the appearance embedding of an image and the structure embedding of the other one. In this context, the generated image should share, for example, the same clothing color of the instance associated with the appearance code and the same pose of that associated with the structure code. To regularize the generator, we can introduce a self-reconstruction loss by feeding the network with the same image or two different images of the same person. The discriminator will try to understand if the images are fake via adversarial loss while the generator is trying to fool him. On top of this, the authors employ an identity classification task for both the original and the generated images, forcing the appearance code of different individuals to stay apart. The identity classification for the generated images is further guided by a *student-teacher* model. Lastly, they also simulate the change of clothing for a given identity, making the network focus on attributes such as body size, hair or presence of a bag and such.

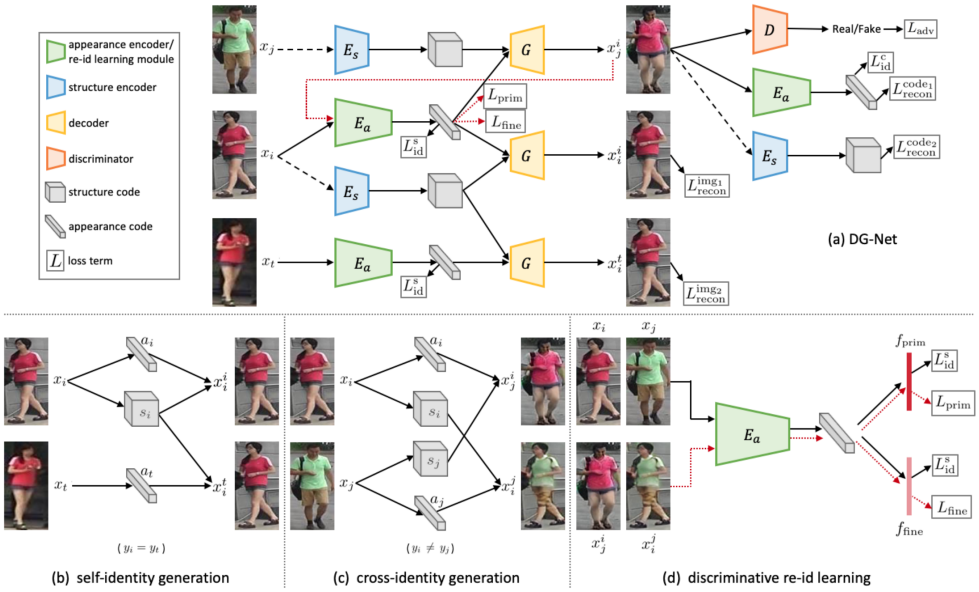

Figure 3.7: DG-Net structure [65]. First row: (a) generative and discriminative modules share the appearance encoder. Second row: (b) self-identity objective with same person, (c) cross-identity objective with two different characters, (d) Re-ID task involves primary and fine-grained feature learning.

3.5 Other approaches

Up until now, we presented works that almost exclusively learn from images (and vector embeddings for pose information in one case). However, it is possible to exploit other kinds of data such as, if provided, segmentation maps and temporal cues. For instance, in [67] the authors extracted both visual semantic and spatial-temporal representations. They argue that it is reasonable to introduce temporal constraints, since when a character has been captured by a certain camera at time t, it cannot be recorded at time $t + \Delta t$ by another camera that is far away from her/him. To do so, they estimated the spatial-temporal distribution of the camera network and exploited the frame numbers available in the data annotations. Having said that, as of today, the majority of approaches focus more on image feature learning rather than exploiting extra information.

Model	**Venue**	Market	
		MaP	R1
PCB[62]	EECV18	81.6	93.8
ST[67]	AAAI18	**95.5**	**98.1**
BOT[61]	CVPR19	94.2	95.4
CBN[64]	ECCV20	83.6	94.3
Top-DB[63]	ICPR20	94.1	95.5
Generative Methods			
FD-Gan[66]	NeurIPS18	77.7	90.5
DG-Net[65]	CVPR19	**86.0**	**94.8**

Table 3.3: Performance on Market.

Model	**Venue**	Duke	
		MaP	R1
PCB[62]	EECV18	69.2	83.3
ST[67]	AAAI18	**92.7**	**94.5**
BOT[61]	CVPR19	89.1	90.3
CBN[64]	ECCV20	70.1	84.8
Top-DB[63]	ICPR20	88.6	90.9
Generative Methods			
FD-Gan[66]	NeurIPS18	64.5	80.0
DG-Net[65]	CVPR19	**74.8**	**86.6**

Table 3.4: Performance on Duke.

Model	**Venue**	CUHK03	
		MaP	R1
PCB[62]	EECV18	57.5	63.7
Top-DB[63]	ICPR20	**86.9**	**85.7**
Generative Methods			
FD-Gan[66]	NeurIPS18	**91.3**	**92.6**

Table 3.5: Performance on CUHK03.

Chapter 4

Cross-Domain Transfer

As we saw in the last chapter, modern supervised person Re-ID systems achieve good performances on most benchmark datasets. However, one might think about how well the previously explained techniques scale in a practical scenario. Before training a model, we would have to collect data coming from our own cameras, annotating manually the identities of all the bounding boxes that we extracted. This job is time-consuming, error-prone and requires a team of more researchers. Along the standard supervised pipelines, there are emerging fields such as *unsupervised*, *semi-supervised domain adaptation* and *domain generalization* for pedestrian Re-ID. Even if supervised methods generalize well to unseen identities (in the test phase), they are not able to retain the same performance when they see images that do not share the same visual characteristics (*e.g.*, illumination, viewpoints, weather conditions, *etc...*) of the training data. In this sense, each dataset could be thought of as belonging to a different *domain*.

This chapter starts with a brief domain adaptation introduction after which there are portrayed several techniques in the realm of deep learning. To highlight the complexity of this task, we point out that, at the moment of writing, these techniques focus mostly on real-to-real datasets. Researchers are trying to bridge the gap between datasets that have been recorded in different scenarios. To conclude the chapter, we finally present existing synth-to-real approaches and introduce our novel synthetic dataset, comparing it to the existing ones.

4.1 Domain adaptation

Suppose that a consulting company has the task to build a reliable pedestrian Re-ID system ready to be deployed in a ship container terminal. Since this is a high-risk area, the port administrators would like to monitor who is entering or exiting the facility. The company does everything by the book, trains a state-of-the-art model

on a benchmark dataset and annotates a smaller one using the port cameras. In the standard supervised training procedure, the identities on which we test, are different from those seen by the model. It is reasonable to expect that, when evaluated on the newly collected samples, the model will still achieve a good performance. Unfortunately, when the company actually tests on the new data, there is a huge drop in accuracy. The consultants then compare the *source* and *target* images, only to see the following differences. In the source dataset the subjects dressed casually, the weather was sunny and the scenes were crowded, while, in the target one, almost all subjects shared the same working outfit, the port was shrouded in mist and there were night scenes. Those two datasets could belong to two different *domains*. What we just described will be framed as *direct transfer* in the next sections, but, before explaining how to address this kind of problem, we formalise common settings of *dataset shift*.

4.1.1 Dataset shift

In a practical scenario, after we trained a model on the available data sources, we would like to get truthful prediction when new data arrives. Unfortunately, the learning and inference environments might change for reasons that range from sample selection bias, measurement error, imbalanced data to more general distribution shifts [68]. This can be modeled by means of conditional and joint probability distributions. Let a domain be the combination of an input feature space $\boldsymbol{X}$ and the probability distribution defined over this feature space $P(\mathbf{x})$, with $\mathbf{x} \in \boldsymbol{X}$. Usually, even if there are different domains, we want to learn a specific task, defined as the combination of the output space $\boldsymbol{Y}$ and the discriminative mapping $\mathbf{x} \mapsto \mathbf{y}$, with $\mathbf{y} \in \boldsymbol{Y}$. This mapping can be expressed as the conditional probability of the output variable (generally the label) given the input one, *i.e.*, $P(\mathbf{y} \mid \mathbf{x})$. In the source-to-target domain adaptation problem, we would like to generalize from the source dataset, to a target one, given that they have some kind of relationship. So far, we have the following:

Source Domain	**Target Domain**
$D^s = \{\boldsymbol{X}^s, P(\mathbf{x}^s)\}$	$D^t = \{\boldsymbol{X}^t, P(\mathbf{x}^t)\}$
$T^s = \{\boldsymbol{Y}^s, P(\mathbf{y}^s \mid \mathbf{x}^s)\}$	$T^t = \{\boldsymbol{Y}^t, P(\mathbf{y}^t \mid \mathbf{x}^t)\}$

where D and T are the domain and task whereas the subscripts s and t respectively indicate source and target. It is possible to write the joint probability distribution for a certain domain [30] as $P(\mathbf{x}, \mathbf{y}) = P(\mathbf{y} \mid \mathbf{x})P(\mathbf{x})$ or $P(\mathbf{x}, \mathbf{y}) = P(\mathbf{x} \mid \mathbf{y})P(\mathbf{y})$. Depending on the assumptions we make about the relationship between source and target, we will end up with different data shift scenarios. We can now present some of the common data shifts [68, 30] that one might incur in.

Covariate shift (figure 4.1) occurs when the probability distribution defined over the source and target feature space differs between those domains, *i.e.*, $P(\mathbf{x}^s) \neq P(\mathbf{x}^t)$. The conditional distributions given the inputs instead, remain equal, *i.e.*, $P(\mathbf{y}^s \mid \mathbf{x}^s) = P(\mathbf{y}^t \mid \mathbf{x}^t)$. To summarize, only the marginal distributions over the feature space change (often called covariates, hence the name) [68], whereas the posterior probability distribution in the target domain must be the same as in the original one.

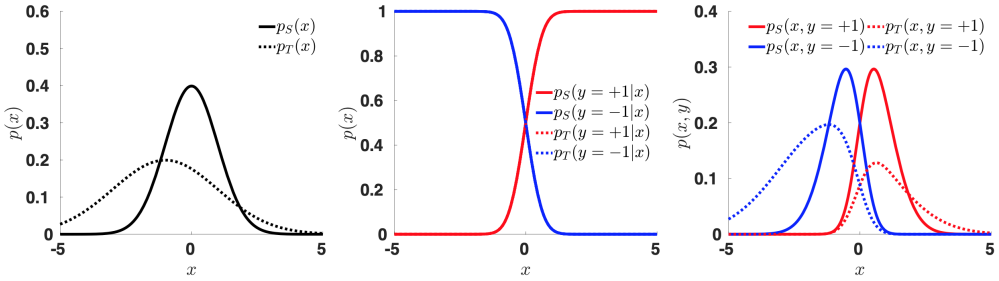

Figure 4.1: Example of covariate shift [30]. Left: the marginal source and target distributions differ. Middle: the posterior distributions coincide. Right: since the target and source distributions are shifted, the joint probability distributions also differ.

When the marginal distributions over the feature space remain the same across the two domains but the posterior (conditional distribution given the data) changes, we have what is called *concept shift* (figure 4.2). Formally, this means that $P(\mathbf{x}^s) = P(\mathbf{x}^t)$, whereas $P(\mathbf{y}^s \mid \mathbf{x}^s) \neq P(\mathbf{y}^t \mid \mathbf{x}^t)$.

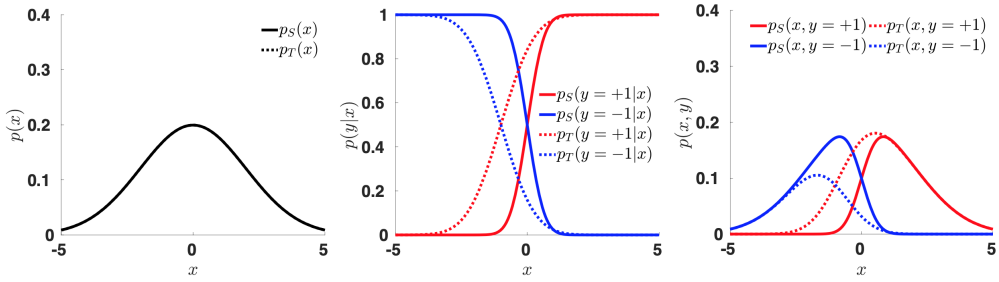

Figure 4.2: Example of concept shift [30]. Left: the marginal source and target marginal distributions are equal. Middle: the posteriors differ. Right: as a consequence, the joint probability distributions also differ.

Lastly, another common dataset shift is characterized by the *prior shift* (figure 4.3). Considering the joint probability decomposition $P(\mathbf{x}, \mathbf{y}) = P(\mathbf{x} \mid \mathbf{y})P(\mathbf{y})$, this data shift refers to the situation where the prior probabilities over the output space

differ between domains but the conditional distributions (given the classes) are equal. That is, $P(\mathbf{y}^s) \neq P(\mathbf{y}^t)$ and $P(\mathbf{x}^s \mid \mathbf{y}^s) = P(\mathbf{x}^t \mid \mathbf{y}^t)$.

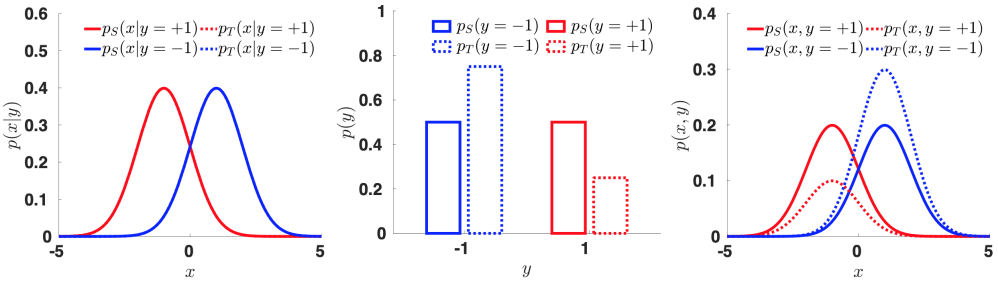

Figure 4.3: Example pf prior shift [30]. Left: the posterior distributions coincide. Middle: the prior of the classes differ between the two domains. Right: the joint probability distributions also differ.

When more general domain shifts occur, we might have a combination of the previous ones. This brings to far more complex scenarios which are difficult to frame theoretically. For instance, in a source-to-target domain adaptation scenario for person Re-ID, usually both the input feature space and class distribution change (since datasets have different identities). Work [69] instead, rethinks the evaluation ranking of the pedestrians as a classification between pairs of images. The output space will be $\boldsymbol{Y} = \{0,1\}$ and we want our model to return 1 when the identities of the pair are the same, 0 otherwise. We can now ask that the labelling function between source and target image pairs should be the same. As of today, there are many approaches to solve this problem.

4.1.2 Deep learning techniques

We will now provide a brief overview of some of the possible deep learning techniques for domain adaptation. Although the literature is vast and there is not a unique general direction, one can expect that certain methodologies will work better than others for certain tasks or that specific approaches could be more relevant in some deep learning sub-fields rather than others. The most straightforward way to address this problem is direct transfer. In this scenario we train a neural network on the source dataset and simply test it on the target one, hoping that our model can make up for the data shift or that the discrepancy among domains is not an issue. Having said that, adopting specific domain adaptation techniques is not guaranteed to increase performance, since the results could depend on the data, task and models of choice. We will mainly focus on what is called *unsupervised domain adaptation* (UDA), where the target domain labels are not available, although the following methodologies can be extended to other settings.

A first relevant technique is that of *deep domain-invariant feature learning* [70]. In practice, we would like to learn robust domain-invariant feature representations aligning the source and target, usually through statistic criterion optimization [71], reconstruction, or adversarial learning [70]. When such feature representation exists, after the learning process, if the classifier has a good performance on the source dataset, it should generalize well to the target domain. Regarding the first option, common statistical criteria consist of minimizing some divergence metrics of the distributions. For instance, the *maximum mean discrepancy* (MMD) measures the difference of the sample mean in a *reproducing kernel Hilbert Space* [72]. Precisely, it is a two-sample statistical test having as null hypothesis the equality of the two feature distribution means. The more they differ, the more likely that the samples belongs to different distributions. This statistic is often used as a distribution discrepancy metric [72]. During the learning process, we can minimize its value to measure the feature alignment (in the next subsection we will provide an example of how MMD could be employed). Regarding reconstruction losses instead, common instances are encoder-decoder architectures. As in work [73], a standard CNN (the encoder) is trained for classification with the source images. At the same time, this backbone shares a low dimensional vector (latent space) with another CNN (the decoder) trained to reconstruct the target images from the output feature vector via some reconstruction losses. The decoder architecture is made up of transposed convolutions such that it gradually upsamples to higher spatial dimensions. As the authors explain, the rationale is that a good domain-adaptive representation (the latent vector in this case) should both allow to correctly classify the source images and reconstruct the target domain data. For adversarial domain adaptation instead, one will usually see a two-player game between a domain and label classifier. These kinds of techniques operate at a feature level, we will later explain generative adversarial methods for domain adaptation. In work [74], a standard CNN is trained for classification with the labeled source images. At the same time, the domain classifier will try to predict the correct domain of both source and target images. In between the last convolutional layer and the domain classifier, there is a gradient reversal layer. During the backpropagation, this layer will invert the sign of the gradient coming from the domain classifier before traversing the feature extractor. As a consequence, the feature extractor will not merely find discriminative features for the downstream task, since it will try to "fool" the domain classifier by learning domain-invariant features. One can imagine this as if the feature extractor and the domain classifier were adversaries. Hopefully, at the end of the training process, the domain classifier will not be capable of distinguishing if an image comes from the source or target domain.

Another set of techniques instead focuses on learning more transferable features by matching (batch) normalization statistics, thus intervening in the network architecture [71]. In [72], the authors argue that batch normalization layers embed

the style and traits of a particular domain. In this sense, they develop a style loss that, during training, aligns the batch normalization statistics of images belonging to two different domains.

In tasks such as person Re-ID, methods that transfer label information with some sort of *class criterion* [71] have gained popularity. Among others, *pseudo labeling* has proven to be effective in guiding the training process. The pseudo labels can be either transferred from the trained source model predictions or via clustering. The latter is common when the source and target do not share the same classes (for instance in retrieval problems). Because of the risk of noisy labels, after a certain number of training steps, pseudo labels are typically recomputed by clustering the newly obtained features, not to stagnate the learning process. In the next section of this chapter, we will provide some examples of this methodology applied to person Re-ID.

Differently from the aforementioned techniques, *domain mapping* [70] typically works on the image pixels level, rather than the feature level. Notable examples are GANs and conditional GANs that *translate* an input image so that it matches the style of the target domain. Although image translation can be thought of as more general than domain adaptation (since it is task agnostic), it is possible to embed other heads into the image mapping architecture to perform specific tasks. For instance, having translated the source images to match the style of the target domain, one could use those mapped images to train a CNN for classification (as done in some person Re-ID works). In this way, we hope to achieve a better performance when we evaluate the target test set as compared to training the CNN on the original source images (direct transfer). Since this approach has heavily influenced our work, in the next subsection we explain some methodologies for image translation in the general context of *neural style transfer*.

On a final note, we spend a few words on *self-supervision* for domain adaptation. Self-supervised methods are characterized by the assignment of automatically generated pseudo labels from data attributes [16]. For instance, we could rotate an image and label it with the rotation angle [75] or divide it into multiple labeled sections, mess up their order and then solve the *jigsaw puzzle* [76]. In practice, we create a new pretext head where, with relevant generated labels, we have to solve a new task while, for instance, jointly training the downstream classification network. The main reason is that, by solving these additional tasks, the network should have a "better understanding" of the objects and their shapes, possibly leading to domain-invariant features. Self-supervision can be thought of as a kind of feature representation learning, particularly useful for domain generalization. In this scenario, there are one or more available source datasets without having a specific target. Self-supervision has successfully been employed for disentangling semantic-specific information from domain-specific information.

4.1.3 Neural style transfer

An interesting setting is when, given a source image and a target one, we would like to retain the content of the former while applying the style of the latter. Neural style transfer can be seen as a disentanglement problem [6], where the objective is to separate the content from the style. Classic examples are picture to painting (usually sketches or impressionist artworks), horse to zebra, cat to dog, *etc...* Remaining in the realm of deep learning, earlier works [77, 8] exploited semantic information from CNN architectures such as the VGG [34] (since it was found to work better in this task than deeper networks) at different layers. For instance, work [8], along with an image transformation network which, given a target image $\mathbf{y}$, maps an input image $\mathbf{x}$ to an output one $\tilde{\mathbf{y}}$, employs two perceptual losses. They match both the content and style of the output image with those of the target one by respectively matching the features maps and the Gram matrix of the feature maps at different layers of a VGG-16 network. A year later, work [72] proved that matching those Gram matrices and minimizing the MMD with the second order-polynomial kernel is equivalent. They also experiment with other kernel functions.

More recent approaches [47, 7, 6] go under the name of image-to-image translation and make extensive use of GANs. Work [47] (*pix2pix*) operates with pairs of images and exploits conditional GANs to map a scene from the source representation to the target one (such as semantic label map to RGB and similar). Although this is a great achievement, in this thesis we are more interested in what is called *unpaired image translation* (see figure 4.4). In this setting, paired instances of both domains are not available, so we need to find a more general mapping to relate the two domains. Among others, *CycleGans* [7] provides an intuitive yet effective framework. Since it was one of the sources of inspiration for our work, we briefly revisit its formulation.

Given the domains $\boldsymbol{X}$ and $\boldsymbol{Y}$ where the instances are drawn according to the data distributions $P(\mathbf{x})$ and $P(\mathbf{y})$, the goal is to learn the mappings $G : \boldsymbol{X} \mapsto \boldsymbol{Y}$ and $F : \boldsymbol{Y} \mapsto \boldsymbol{X}$. This can be modeled with a couple of generators (the mapping functions) and discriminators. Specifically, $D_{\boldsymbol{X}}$ is trained to distinguish between images coming from the domain $\boldsymbol{X}$ and the translated images $F(\mathbf{y})$. The same reasoning applies to the other discriminator $D_{\boldsymbol{Y}}$. One might ask why we need such a complex structure if we want to translate an image from one domain to another. The authors explain that, among other issues (difficulty to optimize the objective, mode collapse), with a standard one-way mapping $G : \boldsymbol{X} \mapsto \boldsymbol{Y}$, even if we could theoretically match the empirical target distribution $P(\mathbf{y})$, it is not guaranteed than each input $\mathbf{x}$ and translated output are meaningfully paired. Given a network with a large enough capacity, input images can be mapped to any random permutation of target images [7]. Since we do not have access to paired source and target images,

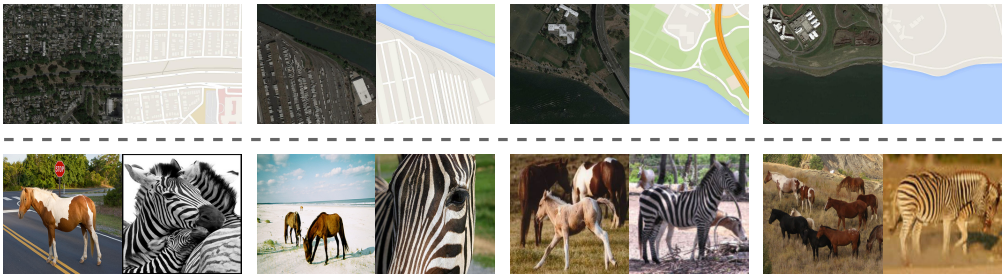

Figure 4.4: Differences between paired and unpaired image translation. First row: image samples [47] are clearly paired and there is a strong relationship between each source and target image. Second row: paired image samples [7] are not available.

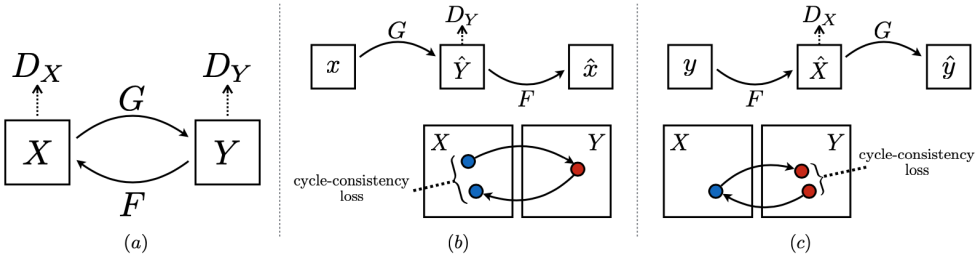

Figure 4.5: Cycle consistency loss [7]. (a) Mapping generators $G : \boldsymbol{X} \mapsto \boldsymbol{Y}$ and $F : \boldsymbol{Y} \mapsto \boldsymbol{X}$ with paired discriminators $D_{\boldsymbol{Y}}$ and $D_{\boldsymbol{X}}$. (b) Forward regularizing cycle consistency loss $\mathbf{x} \to G(\mathbf{x}) \to F(G(\mathbf{x})) \approx \mathbf{x}$. (c) Bacward regularizing cycle consistency loss $\mathbf{y} \to F(\mathbf{y}) \to G(F(\mathbf{y})) \approx \mathbf{y}$. As we can see, after both cycles the reconstructed image should be close to the original one.

we need to give more structure to the objective and guide the generative process. They achieve this by means of the so-called *cycle consistency loss* (see figure 4.5). In this sense, G and F should be the inverse of each other and bijections, that is: $F(G(\mathbf{x})) \approx \mathbf{x}$ and $G(F(\mathbf{y})) \approx y$. By doing so, we are actually constraining the possible obtainable mappings of the generative process. This objective can be expressed as $L_{cyc} = \mathbb{E}_{\mathbf{x}\sim P(\mathbf{x})}\|F(G(\mathbf{x})) - \mathbf{x}\|_1 + \mathbb{E}_{\mathbf{y}\sim P(\mathbf{y})}\|G(F(\mathbf{y})) - \mathbf{y}\|_1$, where we minimize the per-pixel L_1 loss between the original and reconstructed images. For the adversarial loss, instead of using the standard binary cross-entropy loss, the authors employ the L_2 loss for both couples of generator-discriminator. Additionally, to better preserve the color composition of the input image, they regularize the generators by constraining them to return the identity mapping when real instances of the target domain are provided as input. This can be expressed as minimizing the *identity loss*, *i.e.*, $L_{identity}(G, F) = \mathbb{E}_{\mathbf{x}\sim P(\mathbf{x})}\|F(\mathbf{x}) - x\|_1 + \mathbb{E}_{\mathbf{y}\sim P(\mathbf{y})}\|G(\mathbf{y}) - \mathbf{y}\|_1$. Although this framework is effective, work [6] argues that the bijective assumption we make when exploiting the cycle consistency might be too restrictive, since it could

be difficult to reconstruct the samples in both directions. On top of this, training and finding good hyperparameters for a couple of generator-discriminator networks is arduous and time-demanding. The authors propose a one-way translation from source to target with a single generator and discriminator, regularizing the training by comparing patches of the input and output images. Since the overall architecture is lighter than a CycleGan, it is easier to embed secondary tasks and other network heads. From the moment that this represents the core foundation of our work, we will provide more details on the architecture in the next chapter, which will focus on the implementation and methodology details of this thesis.

4.2 Domain adaptation for person Re-ID

After the brief overview on general domain adaptation techniques, it is now possible to focus on the sub-field of person Re-ID. The approaches to bridge the gap across different domains are heterogeneous, with *iterative pseudo-labeling* and generative techniques being among the predominant ones. However, they usually share the same training pipeline. Suppose that we want to generalize from a labeled source dataset to an unlabeled target one. Since pedestrian Re-ID datasets usually have different identities in the training and test splits, the standard procedure is to use, along with the source dataset, the unlabeled training partition of the target dataset to learn domain-related features. The resulting model is then evaluated on the new identities of the target testing partition. In this situation, we are assuming that, in a practical scenario, is far easier to obtain unlabeled data frames instead of labeled ones. Exploring the literature, one will find that such methods are commonly known as UDA [78, 79, 80]. Instead, if we do not exploit any labeled source dataset, learning only from a specific unlabeled "target" dataset, we are in a setting often defined as *fully unsupervised* Re-ID [79]. Other approaches include *semi-supervised learning*, where a small fraction of labeled target data is allowed, and *domain generalization*, where we try to generalize to new unseen domains exploiting more datasets during training or learning domain-agnostic features. In this section, we will explore the state of the art concerning the main approaches for cross-domain (real-to-real) person Re-ID. To see the results of these approaches on the dataset presented in the previous chapter, refer to tables 4.1 and 4.2.

4.2.1 Iterative pseudo-labeling techniques

Work [81] mines similarities among different persons of the unlabeled target dataset by comparing global as well as local information. The authors employed a ResNet-based network that is first trained on a source dataset in a supervised fashion. The model is then fine-tuned on the target dataset by extracting features from the whole images. Each image is then divided into two regions that represent the

upper and lower body parts. They vectorize the feature maps with average pooling and extract pseudo labels from each feature vector via clustering so that every pedestrian is associated with three pseudo labels (for the pedestrian, its lower and upped body parts). The model is then optimized with triplet losses exploiting the obtained feature vectors and pseudo labels (which are iteratively recomputed at each training iteration). As an improvement, they also allow using a restricted sample of manually labeled target images in a semi-supervised training setting.

In the previous example, the pseudo labels were iteratively refined by re-grouping the feature vectors and optimizing for such new pseudo labels at each training iteration. This could be framed as an offline refinery, which, as mentioned in [82], could still lead to noisy labels. Work [82] addresses this exact issue by providing more robust labels with additional online refinery. They employed two CNN with the same architecture (yet different initialization) for collaborative learning. Having extracted the pseudo labels via clustering for the current learning iteration, they optimize one network using as soft labels for the cross-entropy loss the predictions of the other. However, the predictions used as soft labels are averaged temporally so as not to end up with the same result from both networks and reduce the bias due to the noisy nature of the hard pseudo labels. For the evaluation, only the best-performing model is employed. As an improvement of what we have just explained, work [83] modulates the features of the two collaborative networks to further reduce their similarities and bias toward the same error propagation. In each branch, they multiply by a factor a randomly selected block of the feature embeddings, enhancing the differences among the learned parameters. Additionally, to find more discriminative and complementary features, they adopt different kinds of attention mechanisms [35, 36], strategically placing them in the two networks.

One could also try combining more domain adaptation techniques explained in the previous section. For instance, work [84], on top of exploiting unlabeled image clustering, performs adversarial domain invariant feature mapping. The employed CNN tries to fool a domain discriminator to output the same probability (0.5) for both source and target images. Additionally, they try to mitigate the noisy labels with global distance optimization. Usually, inter and intra class feature distances are optimized with contrastive or triplet losses by taking into account only the samples in each batch. However, they combine this kind of local distance optimization with a global one by building a memory bank that contains samples outside the current batch. They found that this strategy enhances the label robustness.

4.2.2 Generative methods

Reviewing the literature, one will find that GANs have been widely adopted to perform different kinds of image translation between pedestrian domains. The main objective is to translate the pedestrian to the target domain while maintaining

their identity-related cues. This will allow us perform the Re-ID task on the domain transferred images, in the hope of obtaining better results compared to a direct transfer. We first present works that exploit the CycleGan framework and subsequently introduce procedures similar to [65] which was presented in the previous chapter in a supervised setting.

The authors of [85] aim to transfer the style of the target images to the source ones while preserving appearance and identity information. Alongside the required losses for the CycleGan, they introduce a novel one that should preserve the identities of the pedestrian during the translation process. They compare the differences between the original and translated images by applying a foreground mask (therefore focusing on the pedestrians) and minimizing the L_2 norm. After this procedure, the result is a translated dataset that allows performing a standard supervised pedestrian Re-ID. In [86], the authors apply a similar framework. However, they also employ a siamese neural network to better preserve identity information of the source images during the translation. Work [78] modifies the original identity constraint of the CycleGan to better align the feature distribution of the same person. However, the authors translate the source dataset only to pre-train a baseline on top of which they perform iterative pseudo labeling techniques. Specifically, they propose a novel *within model co-training* method that provides additional self-supervision during the clustering procedure. As in other works, the learning process is based on minimizing the triplet loss of the features coming from the global average pooling and fully connected layer. However, instead of extracting the pseudo labels only from the features of the last fully connected layer, they perform a second clustering from the global average pooling layer, to obtain another set of pseudo labels. Minimizing the losses coming from different clusterings should make the predictions more reliable since the overall loss will make less confident dissimilar predictions.

Inspired from [65], DG-Net++ [79] disentangles identity-related cues (appearance space) from the unrelated (structure space) ones. It employs a shared appearance encoder for both source and target images and separate structure encoders that model the pose, viewpoint, position, *etc....* Instead of exploiting this disentangling game in a supervised setting, they adopt it to better bridge the gap between the two domains. After generating new images with the appearance of one domain and the structure of the other, they reconstruct the source pictures via cycle consistency (see figure 4.6). The training is also guided by cross-entropy losses on the identities of both domains, using iterative clustering for the unlabeled target images. At the foundation of their motivations is the fact that such disentanglement helps the adaptation process since it constraints to focus on identity-related features only. Simultaneously, adaptation improves features-disentanglement by strengthening the shared appearance encoder.

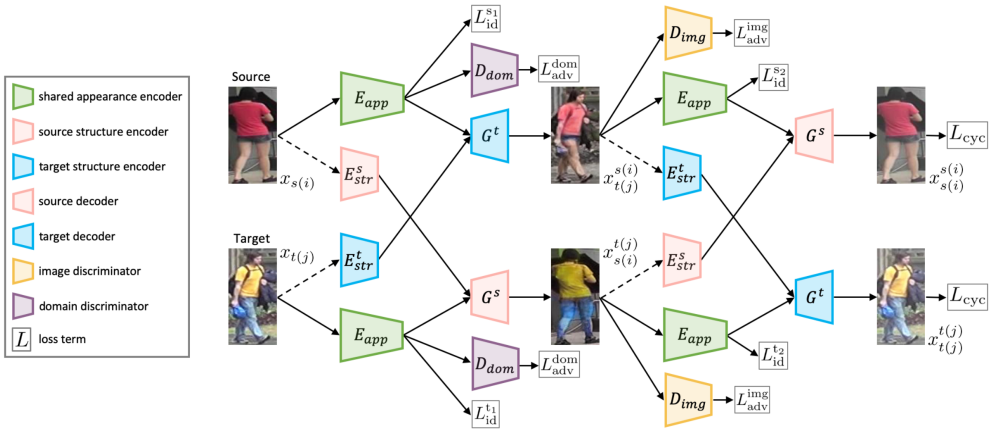

Figure 4.6: Overview of DG-Net++ [79]. It can be seen that the appearance encoder (which is also employed during the evaluation) is shared across the two domains, as well as the image and domain discriminators, whilst the structure encoders are different. After an image is translated given an appearance and structure codes, it is reconstructed back to its original form.

Along the same line of what we just explained, work [80] also develops a disentanglement game between identity-related and unrelated features, but with a core difference. It does not require a source dataset, operating directly with the unlabeled target. For each image, they estimate the pedestrian pose and generate a new one via rotation. The generative module is then asked to synthesize two images starting from the appearance and structure embeddings. While both images will share the identity of the input, one will also share supposedly the same pose (for consistency), whereas the other should appear with the rotated one. This process is enhanced by a contrastive module where the objective is to find, give an anchor image, positive images with the same identity and negative ones with different identities. However, since we do not have access to the pedestrian identities, they extract pseudo labels through clustering.

A different approach is instead that of [87]. Given a source and target images, they first extract identity and contextual cues through two encoders. A mask is applied to the target image for filtering out the person. The information coming from the two pathways is then fused depth-wise and fed into a conditional encoder-decoder GAN. The generated images should resemble the source identity and the background clutter of the target. In particular, the generator returns two outputs, a *residual map* that accounts for domain discrepancy and a *context map* which models context changes (and thus the name *context rendering gan*). The translated image is a combination of the two previous embeddings and the source image. Besides the domain discriminator, they employ cross-camera, inner-camera context

variation and source identity discriminators (figure 4.7).

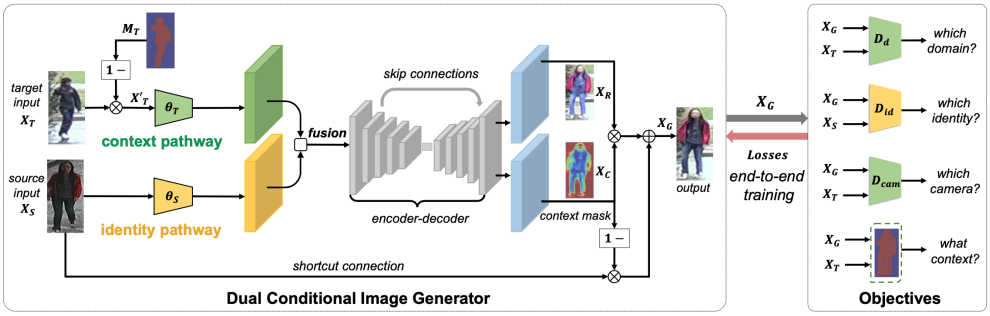

Figure 4.7: CR-GAN [87] (context rendering gan) architecture. The generative process outputs a residual and a context map (respectively $\boldsymbol{X_R}$ and $\boldsymbol{X_C}$) which are combined with the source image to generate the output. On the right side, the different discriminators encoders guide the training together with their corresponding losses.

4.2.3 Other methods

As we saw in the previous two subsections, domain adaptive approaches for person Re-ID rely on a labeled source dataset and a specific unlabeled target one. In the realm of pseudo labeling techniques, the source is often used only for pre-trainings, whereas generative methods leverage and exploit more the source data (although we provided an example where this is not true). On top of this, we often assume to have a decent amount of unlabeled target images, since most approaches bridge the gap from a specific source to a specific target. As a consequence, in a practical scenario, we would need to perform domain adaptation for multiple locations if they differ enough to drop the performance. Domain generalization techniques try to counter this issue. For instance, work [5] employs an architecture with several convolutional streams, each of them having a different receptive field size. This will constrain the network to look for multiple discriminative regions with different scales for the same image, capturing more details and potentially reducing the false matches. The streams are then fused together with an aggregating gate that will assign more weight on one or more scales. The model is optimized in a supervised fashion, exploiting the source dataset.

Up until now, all the approaches that we detailed only focused on visual cues, as the majority of person Re-ID works. However, in the previous chapter, we saw that one could also exploit different kinds of information. A good example can be found in [88], where the authors also make use of temporal information. To summarise, they assume that, because of the relatively small size of each batch of data, images

are likely to have distinct identities inside a batch. Therefore, they optimize on a local level with a classification model that pushes the pedestrians away from each other. On a global level (considering the whole training data), instead of just considering only visual features to estimate the pseudo labels, they also rely on the available video frame numbers for temporal consistency. This temporal consistency is based on how likely an individual is to be captured by other cameras in a future time instant, given that he/she is being recorded by a certain camera right now. They then build temporal-guided clusters [88] for label prediction combining such temporal consistency and visual similarities.

Model	**Venue**	**Source**	Market	
			MaP	R1
SSG[81]	ICCV19	Duke	58.3	80.0
SSG++[81]	ICCV19	Duke	68.7	86.2
DIM+GLO[84]	ACMM20	Duke	65.1	88.3
MMT[82]	ICLR20	Duke	76.5	91.2
JVTC+[88]	EECV20	Duke	67.2	86.8
OSNet-AIN[77]	IEEE-TPAMI21	Duke	30.6	61.0
OSNet-AIN[77]	IEEE-TPAMI21	MSMT17	43.3	70.1
AWB[83]	IEEE-TIP22	Duke	**81.0**	**93.5**
AWB[83]	IEEE-TIP22	MSMT17	79.4	92.6
Generative Methods				
SPGAN[86]	CVPR18	Duke	26.9	58.1
Cyclegan[78]	CVPR19	Duke	24.5	52.0
ISSDA[78]	CVPR19	Duke	63.1	81.3
CR-GAN[87]	IICV19	Duke	33.2	64.5
CR-GAN[87]	IICV19	CUHK03	30.4	58.5
CR-GAN+TAUDL[87]	IICV19	Duke	54.0	77.7
CR-GAN+TAUDL[87]	IICV19	CUHK03	56.0	78.3
DG-Net++[79]	EECV20	Duke	61.7	82.1
DG-Net++[79]	EECV20	MSMT17	64.6	83.1
GLC[80]	CVPR21	None	66.8	87.3
GLC[80]	CVPR21	Duke	**75.4**	**90.5**

Table 4.1: Table with models performances when trained on a source dataset and tested on Market. To notice that SSG++ allows a small subset of the target labels (semi-supervised learning) and that OSNet is target agnostic (domain generalization).

Model	**Venue**	**Source**	Duke	
			MaP	R1
SSG[81]	ICCV19	Market	53.4	73.0
SSG++[81]	ICCV19	Market	60.3	76.0
DIM+GLO[84]	ACMM20	Market	58.3	76.2
JVTC+[88]	EECV20	Market	66.5	80.4
MMT[82]	ICLR20	Market	68.7	81.8
OSNet-AIN[77]	IEEE-TPAMI21	Market	30.5	52.4
OSNet-AIN[77]	IEEE-TPAMI21	MSMT17	52.7	71.1
AWB[83]	IEEE-TIP22	Market	**70.9**	**83.3**
AWB[83]	IEEE-TIP22	MSMT17	69.6	82.8
Generative Methods				
SPGAN[86]	CVPR18	Market	26.4	46.9
Cyclegan[78]	CVPR19	Market	19.4	35.7
ISSDA[78]	CVPR19	Market	54.1	72.8
CR-GAN[87]	IICV19	Market	33.3	56.0
CR-GAN[87]	IICV19	CUHK03	26.9	46.5
CR-GAN+TAUDL[87]	IICV19	Market	48.6	68.9
CR-GAN+TAUDL[87]	IICV19	CUHK03	47.7	67.7
DG-Net++[79]	EECV20	Market	63.8	78.9
DG-Net++[79]	EECV20	MSMT17	58.2	75.2
GLC[80]	CVPR21	None	62.8	**82.9**
GLC[80]	CVPR21	Market	**67.6** 81.9	

Table 4.2: Table with models performances when trained on a source dataset and tested on Duke. To notice that SSG++ allows a small subset of the target labels (semi-supervised learning) and that OSNet is target agnostic (domain generalization).

4.3 Synthetic datasets

Although domain transfer techniques have been dramatically evolving in this field, there seems to be a new emerging direction. As we pointed out in the previous chapter, large-scale datasets are challenging to record and annotate [89]. Furthermore, illumination, pose and viewpoint variations are limited by the physical world and camera availability, lowering the performance of a model across domains. From an ethical standpoint instead, the acquisition of such datasets in public environments has raised privacy concerns [90]. To address these issues, recent works make use of software-generated data, both in direct transfer and domain adaptation settings. Below we introduce some of the existing synthetic datasets

for person Re-ID and compare them to ours. Table 4.3 provides some summary statistics.

4.3.1 SyRI

In this dataset [91] we can find 100 digital pedestrians with custom body shapes, clothing and accessories. The authors used Unreal Engine 4 [92] to animate the characters with gender-dependent walking rigs. For the illumination, they used 140 HDR environmental maps, where each of them can be seen as a sphere surrounding the pedestrian with certain lighting conditions. Each character has been recorded with all the HDR maps and two random rotations along the vertical axis, for a total of 1,680,000 extracted frames.
This dataset (examples in figure 4.8) was built to support the author's work of illumination inference. In fact, it appears much different from real-world datasets since it does not present a multi-camera network, surveillance scenes and pedestrian occlusions.

Figure 4.8: Image samples from SyRI [91]. They depict the same character in different HDR maps.

4.3.2 PersonX

This dataset [93] was built with Unity3D [94] and presents 1,266 different characters, 547 females and 719 males. To increase realism, the authors created materials and textures from mappings obtained by scanning real-world people. The pedestrians' diversity is assured by different skin colors, ages, body shapes, hairstyles and other clothing accessories. They present different illumination sources such as spotlight, area light and sunlight with editable parameters. The virtual cameras present configurable values for image resolution, focal length and height. Camera views are tied with 6 different backgrounds, where the characters can be animated in different directions. Each pedestrian has been recorded from 36 different angles in each background, for a total of 273,456 image samples (examples in figure 4.9).

4.3.3 RandPerson

The authors of this dataset [95] used the software MakeHuman [96] to generate 8,000 different 3D pedestrians. To differentiate the synthetic identities, they generated

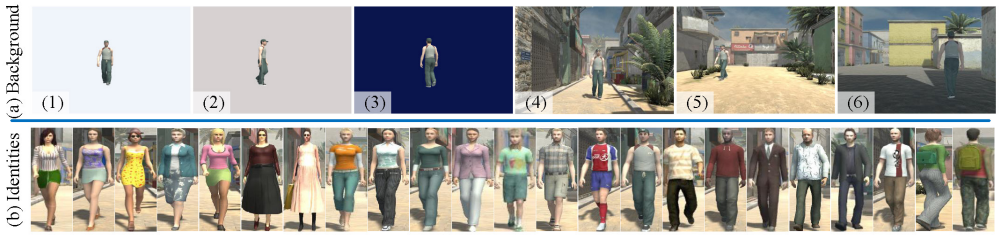

Figure 4.9: PersonX[93] dataset. First row (a): pedestrian captured in different views and backgrounds (1-3 are uniform color backgrounds while 4-6 have scenes). Second row (b): bounding box samples of different pedestrians with various clothes.

new outfits by creating new UV texture maps. There is an equal gender and skin color distribution, weights and ages are uniformly distributed in the software ranges whereas they used a normal distribution for the heights. Each character is further customized with other accessories such as beards, shoes, necklaces and hairstyles. To simulate real video surveillance scenes, they created and modified several virtual environments with Unity3D [94]. Pedestrians are recorded from multiple cameras, varying illumination, viewpoint distance, resolution and background (see figure 4.10). Additionally, person-to-person occlusions are common such as in real-world scenarios. There are 19 cameras recording a total of 11 scenes. From the 38 recorded video, they extracted 1,801,816 images.
Among others, the main features of this dataset that better distinguish it from previous synthetic ones (such as SOMAset [97], SyRI [91] and PersonX [93]) are the presence of multiple cameras and pedestrians in the same scene. Instead of just capturing a pedestrian with a single camera from multiple viewpoints, they created proper virtual Re-ID scenes with more pedestrians and cameras at once.

Figure 4.10: Image sample of the RandPerson dataset [95]. First row: same character in different scenes. Second row: different characters in the same scene.

4.3.4 UnrealPerson

This dataset [89] share some similarities with the previous one, such as using MakeHuman to generate pedestrians and design virtual person Re-ID scenes. However, the authors used real-world clothing images to generate cloth textures to increase realism. Additionally, they also deliberately included pedestrians with similar characteristics to add "hard samples". Although having a diverse dataset in terms of accessories, clothes, identities is important, similar pedestrians play an important role since this could guide Re-ID systems to focus on discriminative details [89]. Overall, they also claim that the generated surveillance environments are more realistic than those of previous datasets.
The scenes were simulated with the software Unreal Engine 4 [92], comprising of 3 outdoors and 1 indoor environments. As in the RandPerson dataset, there are person-to-person occlusions and illumination variations. They extracted 120,000 bounding boxes captured over 34 different cameras, depicting 3,000 digital identities.

4.3.5 GTASynthReid

We will now provide the main characteristics of our dataset and explain how it was created. Instead of relying on 3D software for building our own scenes and pedestrian models, we exploited the game engine of *Grand Theft Auto V* (GTAV) [98]. This game has increasingly become more popular in the computer vision community for its photorealism and adaptability to satisfy different scenarios. Over the years, several user-generated scripts (*mods*[1]) have been made available to the public with the aim of modifying the original game mechanics and transforming it into a development tool. In the literature, we can see that GTAV has been used for pedestrian Re-ID [90], path prediction [99], photorealism enhancement [100] and other tasks related to vehicles (*e.g* vehicle detection, path prediction, *etc...*). The main rationales behind the employment of this game are, besides the realistic-looking scenes, the capability to program vehicles, pedestrians, weather conditions and the presence of already made 3D environments and models. Additionally, it is not that difficult to extract data such as segmentation regions, pedestrian coordinates and paths, other than images or videos.

In our work, we exploited GTAV to create a synthetic dataset for pedestrian Re-ID. This was done by modifying some preexisting mods[2] developed in [99] for

[1]The term "mod" in the video games context refers to (user-generated) scripts that *modify* the regular game storytelling, functionality and environments.

[2]In this work we tailored the mods available at https://github.com/fabbrimatteo/JTA-Mods for our task. We also modified the scripts available at https://github.com/fabbrimatteo/JTA-Dataset needed to extract the bounding boxes from the video frames. Both software repositories

pedestrian pose estimation and tracking in urban scenarios. In practice, we first selected the 538 in-game available characters that were appropriate for this task (excluding other non-human characters) and identified some interesting security locations. Then, we programmed the pedestrians to follow a certain path or roam around an area and we recorded the scenes. Each location can have up to two cameras (in that case the frame rate is doubled to record one frame for each camera sequentially) and present crowded environments, where pedestrians can also collide. The characters were captured in several urban scenarios with a total of 19 different camera positions (both overlapping and non-overlapping). Each pedestrian has been recorded by at least 2 camera views with an average of 3.5, up to 5 cameras. There are scenes both at day-time and night-time, with different kinds of weather (blizzard, rain, clear). The pedestrians have been recorded with different camera views, illumination and weather settings (see figure 4.11 for some examples). The assumption is that each identity maintains the same clothing in all the videos. The total number of bounding boxes is 94312 and it ranges from 29 to 496 for each pedestrian. To increase realism, the bounding boxes are not perfect, meaning that the characters could be cropped as well as occluded by objects or other pedestrians. Since the scenes are crowded, there are person-to-person interactions, leading to frames where other pedestrians (or body parts) are present besides the person of interest. Additionally, since there are characters who wear uniforms or suits, some of the individuals share similar clothes, even though they represent different identities. This could be linked also to scenes with low illumination or when the subjects are far away from the camera. For instance, when images are "pixelated", is more difficult to differentiate among characters, as in real-world datasets. Although our dataset might feel more realistic compared to similar ones, for the identities we were limited to only those allowed by the game.

Dataset	**#Cameras**	**#Scenes**	**#Bboxes**	**#Identities**
SyRI[91]	280	140	1,680,000	100
PersonX[93]	6	1	273,456	1,266
RandPerson[95]	19	11	1,801,816	8,000
UnrealPerson[89]	34	4	120,000	3,000
GTASynthReid	19	5	94,312	538

Table 4.3: Summary of the main synthetic datasets characteristics. Although we provide the total number of bounding boxes, we notice that it is often undersampled.

were developed in [99].

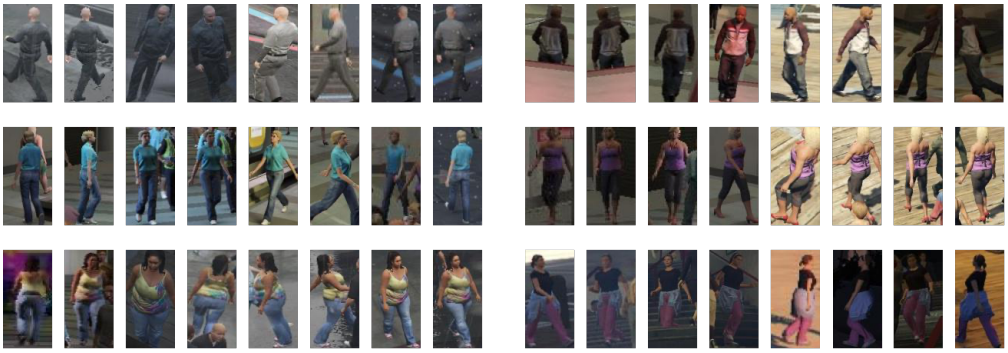

Figure 4.11: Frames from our synthetic dataset. There are 6 pedestrians, each depicted in 8 different bounding boxes. One can notice different viewpoints, illumination settings, pose and weather conditions. There are also occlusions and person-to-person interactions (other pedestrians or body parts).

4.4 Synth to real

In the last section of this chapter, we present different methodologies that have been implemented to generalize from synthetic to real data. In the context of person Re-ID, this is a relatively recent field of research that, at the time of writing, is rapidly evolving. Having presented some of the existing computer-generated datasets for pedestrian Re-ID, we also provide the results of these novel approaches on such datasets (see tables 4.4, 4.5 and 4.6).

4.4.1 Approaches

An interesting earlier approach is that of [97]. In this work, the authors created a novel synthetic dataset, SOMAset, with 25 male and 25 female characters. The pedestrians present different somatotypes with different body shapes and, additionally, each of them has been depicted with 8 different sets of clothing. This is different from the vast majority of person Re-ID datasets, since researchers often assume that each identity has the same clothing across different scenes. They also account for ethnicity variations. For the image frames, each pedestrian has been recorded with 250 poses for each set of clothing. For the Re-ID task, they develop an Inception-based architecture [41] which is trained on their synthetic dataset. To account for synth-to-real domain variations, the network is then fine-tuned on the real target dataset.

Work [91] addresses the synth-to-real domain adaptation in a three-step process. At first, the authors perform illumination inference exploiting the 140 environmental maps of the SyRI dataset. In practice, they consider each illumination condition as a different domain and train a neural network to predict the illumination condition

of each synthetic image. Then, they find the closest lighting condition of the real target dataset by majority voting. The rationale is to achieve the minimum domain shift by considering only the synthetic lighting setting which is the closest to the real target data. The chosen synthetic domain is then adopted in the second step for the domain translation. They employed a cyclegan similarly to what we explained in the neural style transfer subsection 4.1.3 (without needing the target training labels). Finally, they fine-tuned CNN with the translated images for the task of person Re-ID.

Recent works [89, 95] focus more on creating a detailed synthetic dataset rather than developing ad-hoc models for synth-to-real adaptation. For instance, in [95] the authors performed multiple experiments by simply employing a backbone ResNet50 [4] for direct transfer. They show that with their RandPerson dataset they achieve better performance with respect to other synthetic datasets (PersonX, SyrRI, SomaSet), even surpassing some of the previous real-to-real domain adaptation techniques. Similarly, the authors of [92] tested their UnrealPerson with various methods against real datasets. However, instead of just using a ResNet, they employed more advanced techniques. After doing several experiments with BoT[61] and CBN [64] for direct transfer, they also employed JVTC [88] for domain adaptation. Not only do they surpass some recent real-to-real domain adaptation works, but they also obtain results close to supervised settings.

As we just explained, modern synthetic person Re-ID systems heavily rely on bigger and more detailed datasets instead of just developing new deep learning models. In this sense, we indeed created a new dataset exploiting the game engine of GTAV but we also developed a generative adversarial model. During the learning process, we tried to increase the realism of the synthetic images by translation while jointly training a model for the Re-ID task. Apart from training a successful model for this kind of task, we think that it is also meaningful to investigate synth-to-real image translation. In the following chapter, we will unveil our architecture and explain the obtained results.

Model	**Venue**	**Source**	Market	
			MaP	R1
SyRI[91]	EECV18	SyRI	-	65.7
Resnet50[95][4]	ACMM20	RandPerson	28.8	55.6
CBN[89][64]	CVPR21	UnrealPerson	54.3	79.0
BoT[89][61]	CVPR21	UnrealPerson	37.2	64.0
JVTC[89][88]	CVPR21	UnrealPerson	**80.2**	**93.0**

Table 4.4: Performances of the explained synth-to-real techniques tested on Market.

Model	**Venue**	**Source**	Duke	
			MaP	R1
Resnet50[95][4]	ACMM20	RandPerson	27.1	47.6
CBN[89][64]	CVPR21	UnrealPerson	49.4	69.7
BoT[89][25]	CVPR21	UnrealPerson	37.5	58.0
JVTC[89][88]	CVPR21	UnrealPerson	**75.2**	**88.3**

Table 4.5: Performances of the explained synth-to-real techniques tested on Duke.

Model	**Venue**	**Source**	CUHK03	
			MaP	R1
Resnet50[95][4]	ACMM20	RandPerson	10.8	13.4

Table 4.6: Performances of the explained synth-to-real techniques tested on CUHK03.

Chapter 5

Model Architecture and Experiments

In our work, we intended to combine discriminative feature learning with image translation. The former paradigm is directed towards the Re-ID task, while the latter should bridge the gap between two or more domains. In this sense, we fused the translation and feature extraction modules into one stage so that the latter can benefit from the former. The translation module should make the source images style more similar to the target one while the feature extractor learns pedestrian descriptors by exploiting transferred features.

In this chapter, we will finally unveil the architecture of our approach and describe the obtained results. We first provide a detailed overview of the model, revisiting the original inspirations. Then, we point out the training pipeline with the selected hyperparameters and reveal the related technicalities. After that, we provide our results and compare them to existing techniques.

5.1 Network Architecture

As we mentioned in the previous chapter, we adopted the framework of [6] for the image translation. To the best of our knowledge, this framework has not yet been introduced in other pedestrian Re-ID works. The authors propose a *contrastive unpaired translation* (CUT) which employs only a single generator and discriminator. Since it is difficult to reconstruct the image samples in both directions, they give up on the bijective assumption of CycleGans and instead constrain the generative process by comparing patches of the input and generated images. Given a pair of source and target pictures, the aim resides in generating an output with the same content of the source image (in our case a specific identity) while matching the general appearance (*e.g.*, style, texture) of the target image. They portray

this as a disentanglement problem, separating the domain-invariant content of an image from its appearance. Precisely, when comparing patches of the input and its translated output, corresponding patches should relate, preserving the original relationships. The generator of the translation module has an encoder-decoder structure (see figure 5.1). Since the encoder learns features responsible for the translation, we exploit those features as a low-level image-translated descriptor. This embedding is then fed into a Resnet-like network to perform the Re-ID task.

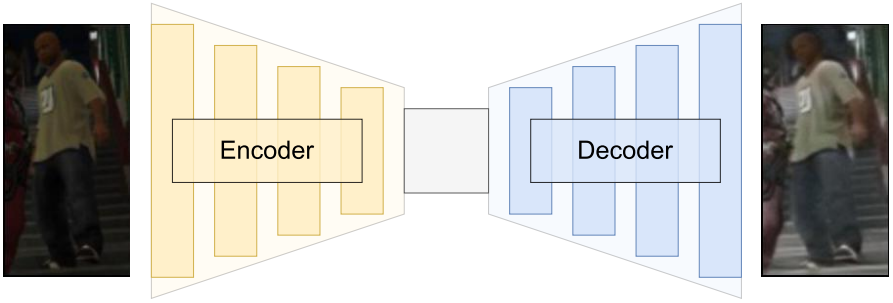

Figure 5.1: Encoder-Decoder structure of the generator in CUT [6]. The encoder learns a lower representation (in the gray painted latent code) of the input image, whilst the decoder learns to build an image starting from that lower representation. Combined with the discriminator, this structure will generate an output that matches the target style (Market [1] target dataset in this case) while retaining the content of the source picture.

5.1.1 Domain mapping

Given our synthetic labeled source dataset $\boldsymbol{X}$, we want to translate it so that its images appear like those of a target unlabeled real dataset $\boldsymbol{Y}$. As noted in [6], we learn a one-way, synth-to-real mapping exploiting GANs and adversarial training. However, there are two main differences from what we explained in chapter 2. First of all, the generator G does not sample from a random noise prior. Instead, it shares an encoder-decoder structure (see figure 5.1). When it receives a batch of synthetic images, the generator's encoder first learns a low dimensional representation of the images themselves, then, the decoder generates the output images starting from those feature vectors. To summarize, the authors state that an output image $\hat{\mathbf{y}}$ is generated as follows: $\hat{\mathbf{y}} = G(\mathbf{x}) = G_{dec}(G_{enc}(\mathbf{x}))$, where $\mathbf{x} \in \boldsymbol{X}$. In practical terms, this is done to preserve the semantics and general appearance of the source images since we want to generate a more realistic version of them. The other important difference relies in the structure of the discriminator D. Instead of employing a fully connected layer as the classifier, they used *PatchGan* [47]. The classifier is replaced by a convolutional layer with a channel size equal to 1. Each spatial

location of the final feature map looks at a certain region in the input image. In this case, through receptive field arithmetic, one can see that each patch in the original image is a 70×70 region. Thus, instead of classifying an entire image as real or fake, we classify whether each patch of an image is real or fake. Work [47] states that this procedure should alleviate from artifacts and promote sharp (colorful, less blurry) outputs.

Regarding the optimization and losses, this problem can be framed as a min-max game with value function $V(G, D)$:

$$\min_G \max_D V(G, D) = \mathbb{E}_{\mathbf{y} \sim \boldsymbol{Y}} \Big[\log D(\mathbf{y}) \Big] + \mathbb{E}_{\mathbf{x} \sim \boldsymbol{X}} \Big[\log(1 - D(G(\mathbf{x}))) \Big] \tag{5.1}$$

where $\mathbf{y}$ and $\mathbf{x}$ are images respectively sampled from the real and synthetic datasets. The main difference with what we explained in chapter 2 is that the generator is no longer sampling random vectors from the latent space but images from the synthetic data. In practice, we used the MSE loss because of its advantages, as the authors suggest:

$$\mathcal{L}^D(G, D, \boldsymbol{X}, \boldsymbol{Y}) = \mathbb{E}_{\mathbf{y} \sim \boldsymbol{Y}} \Big[(D(\mathbf{y}) - 1)^2 \Big] + \mathbb{E}_{\mathbf{x} \sim \boldsymbol{X}} \Big[(D(G(\mathbf{x})))^2 \Big] \tag{5.2}$$

$$\mathcal{L}^G(G, D, \boldsymbol{X}) = \mathbb{E}_{\mathbf{x} \sim \boldsymbol{X}} \Big[(D(G(\mathbf{x})) - 1)^2 \Big] \tag{5.3}$$

In the realm of GANs, there has been lots of research on how to improve the generator's architecture. Inspired by works [8, 77] where they compare feature maps at different layers, we instead introduce a *feature matching similarity* loss for the discriminator. We argue that, since the discriminator is a simple CNN with just few layers, it should capture general concepts of the images it sees. From the discriminator's perspective, we ask that the resulting feature maps of a certain layer should be similar when the discriminator itself is fed with a synthetic (source) and a translated image. Nevertheless, when dealing with adversarial training, one should think in both players' perspectives. Thus, when optimizing for the generator, we also ask that the feature maps for a specific discriminator's layer should be similar when the latter is fed a translated and a target image. The similarity is measured by means of MSE. These two criteria can be written as:

$$\mathcal{L}^D_{Sim}(G, D, \boldsymbol{X}) = \mathbb{E}_{\mathbf{x} \sim \boldsymbol{X}} \frac{1}{L} \sum_{l=1}^{L} \Big(D_l(\mathbf{x}) - D_l(G(\mathbf{x})) \Big)^2 \tag{5.4}$$

$$\mathcal{L}^G_{Sim}(G, D, \boldsymbol{X}, \boldsymbol{Y}) = \mathbb{E}_{(\mathbf{x}, \mathbf{y}) \sim (\boldsymbol{X}, \boldsymbol{Y})} \frac{1}{L} \sum_{l=1}^{L} \Big(D_l(\mathbf{y}) - D_l(G(\mathbf{x})) \Big)^2 \tag{5.5}$$

where D_l, with $l \in 1, 2, ..., L$, represents the output feature map of the l^{th} discriminator's layer and $G(\mathbf{x})$ is a translated image. For the generator's similarity loss, we sample a tuple $(\mathbf{x}, \mathbf{y})$ containing samples from both the source and target datasets. We can now express the overall gan loss as:

$$\mathcal{L}_{Gan}(G, D, \boldsymbol{X}, \boldsymbol{Y}) = w_D \mathcal{L}^D(G, D, \boldsymbol{X}, \boldsymbol{Y}) + w_G \mathcal{L}^G(G, D, \boldsymbol{X}) + \\ + w_{SD} \mathcal{L}^D_{sim}(G, D, \boldsymbol{X}) + w_{SG} \mathcal{L}^G_{Sim}(G, D, \boldsymbol{X}, \boldsymbol{Y}) \tag{5.6}$$

where w_D and w_G balance the discriminator and generator contribution to the adversarial loss, whilst w_{SD} and w_{SG} are the weights for the discriminator and generator similarity losses.

Figure 5.2: Pair of images where the sources (left) are translated to match the Market dataset style. By using a simple generator-discriminator architecture, the outputs (right) do not hold any translation or content value. This shows how important is to better guide and constrain this process.

5.1.2 Relationship preservation

Learning the previous mapping alone is not enough to generate realistic versions of our synthetic pedestrians. The results will present hallucinations, artifacts, or will not even resemble human beings (see figure 5.2). The previous mapping is too generic, especially since we are considering unpaired source and target images. To alleviate these problems, CUT [6] guides the generative process with additional constraints. It can be seen as an alternative to the bijective mapping of cycle consistency. Intuitively, after the translation, one would like the corresponding regions in the input and translated images only to differ in style. This means that we want to preserve the input content (a leg should still look like a leg, the same for arms and faces). They do this by comparing input and output image patches, directing the optimization process towards more relevant outputs by applying a noise contrastive estimation framework [101] to the image translation. This framework has been designed for unsupervised representation learning, aiming at generating compact latent embeddings from high-dimensional data by encoding the shared information between different parts of the training data, as explained in [101]. In our setting, we would like to maximize the mutual information between the input and output images. Given such an input-output pair, we can think of both images as a collection of several patches (smaller image regions). Following

the notation of [6], in a contrastive learning environment, we have to associate the query (*e.g.*, an output image patch) with the positive sample (the corresponding patch of the input image) against all the negative ones (all the other input image patches). The query, positive and N negatives are mapped respectively to the K-dimensional feature vectors (with a neural network) $\mathbf{v} \in \mathbb{R}^K$, $\mathbf{v}^+ \in \mathbb{R}^K$ and $\mathbf{v}^- \in \mathbb{R}^{N \times K}$. Now that we have these ingredients, it is reasonable to construct some kind of loss that will penalize us when the output and input patches are not associated. After having normalized to a unit sphere all the feature vectors, we can compute the dot product to measure the *cosine similarity* between a pair of vectorized patches. In the CUT framework, the authors adapted the *InfoNCE* loss of [101] to guide the generative process by asking for high similarity when matching corresponding patches. Intuitively, this can be thought of as a $N+1$ classification problem, where the cross-entropy loss is defined as:

$$\mathcal{L}_{InfoNCE}(\mathbf{v}, \mathbf{v}^+, \mathbf{v}^-) = -\log \left[\frac{\exp(\mathbf{v} \cdot \mathbf{v}^+ / \tau)}{\exp(\mathbf{v} \cdot \mathbf{v}^+ / \tau) + \sum_{i=1}^{N} \exp(\mathbf{v} \cdot \mathbf{v}_i^- / \tau)} \right] \tag{5.7}$$

where $\mathbf{v}_i^- \in \mathbf{v}^-$ is one of the negatives (not associated with the query) input patches and the dots products are scaled by the temperature factor $\tau = 0.07$. Minimizing the previous expression will help constrain the generative process in producing images that better preserve the input relationships.

We can now understand how to fit together the previous framework with a GAN. As it was mentioned above, we need to divide the images into patches and subsequently map them to K-dimensional feature vectors. However, this is already partly done by the generative module. If we think about it, when an image is fed to a CNN, each neuron activation of the output feature map of a certain layer looks at a region in the original image. The deeper we are, the bigger the region we are looking at will be (receptive field arithmetic). To this extent, CUT exploits the generator's encoder G_{enc} to extract embeddings of the image patches (see figure 5.3 for an example). Furthermore, we can select feature maps at different depths and optimize eq: 5.7 with each set of vector patches, repeating the process for different patch sizes. Up until now, we have that, for each selected layer l of G_{enc}, there are S_l spatial locations ($width \times height$ of the feature map) looking at as many input patches represented by C-dimensional features vectors, where C is the channel size of layer l. Work [6] further processes the feature maps of the L layers of interest by passing them through a two-layer MLP H_l, with $l \in \{1, 2, ..., L\}$. It is crucial to notice that there is not a single MLP, but rather one for each of the selected layers. These smaller MLPs can be thought of as layer-specialized neural networks for the contrastive task. When put all together, we have a set of L new feature maps: $\{\mathbf{z}_l\}_L = \{H_l(G_{enc}^l(\mathbf{x}))\}_L$, where $G_{enc}^l(\mathbf{x})$ is the output feature map of the l^{th} chosen encoder layer. For a selected layer l of G_{enc}, $\mathbf{z}_l^s \in \mathbb{R}^{C_l}$ is the feature vector

associated to the spatial location $s \in S = \{1, 2, ..., S_l\}$ of the feature map, where C_l now depends on the architecture of both G^l_{enc} and H_l. Since the query comes from the generated images $\hat{\mathbf{y}} = G(\mathbf{x})$, given an output image we repeat the same procedure: $\{\hat{\mathbf{z}}_l\}_L = \{H_l(G^l_{enc}(G(\mathbf{x})))\}_L$. As shown in [6], this can be expressed by what they call the *PatchNCE* loss:

$$\mathcal{L}_{PatchNCE}(G, H, \boldsymbol{X}) = \mathbb{E}_{\mathbf{x}\sim\boldsymbol{X}} \sum_{l=1}^{L} \sum_{s=1}^{S_l} L_{InfoNCE}\left(\hat{\mathbf{z}}^s_l, \mathbf{z}^s_l, \mathbf{z}^{S/s}_l\right) \tag{5.8}$$

where H is the set of all the MLPs, $\hat{\mathbf{z}}^s_l$ is the query coming from the output image, $\mathbf{z}^s_l$ is the positive sample of the input image and $\mathbf{z}^{S/s}_l \in \mathbb{R}^{(S_l-1)\times C_l}$ are all the input negatives. The negative patches could be sampled also from the rest of the source dataset. However, the authors of CUT find out through experiments that internal (same input image of the positive sample) patches outperform the external ones. We followed their direction and employed only internal patches. In the CycleGan [7] framework, the image translation is further constrained by the so-called identity loss (to read as identity mapping, not intended as the identity of a pedestrian). As we explained in the previous chapter, this loss should help retain pedestrian characteristics of the input image, penalizing major changes. Hence, we need to feed the generator a target image and expect the identity mapping for the result. In CUT this concept has been revisited, exploiting the PatchNCE loss instead of the MSE:

$$\mathcal{L}_{PatchNCE}(G, H, \boldsymbol{Y}) = \mathbb{E}_{\mathbf{y}\sim\boldsymbol{Y}} \sum_{l=1}^{L} \sum_{s=1}^{S_l} L_{InfoNCE}\left(\hat{\mathbf{z}}^s_l, \mathbf{z}^s_l, \mathbf{z}^{S/s}_l\right) \tag{5.9}$$

where the only difference from eq. 5.7 is that instead of passing images from the source dataset X, we take them from Y, the target one.

5.1.3 Discriminative learning

A natural way of performing the Re-ID task would be to train a neural network on the translated dataset, hoping that it will perform better than a direct transfer on the target data, as in [78, 86]. However, there is an increasing trend of embedding the discriminative module. As in [79, 80], we argue that feature learning could benefit from the domain adaptation process. In our context, integrating a network for classification with the translation module will expose the generator's domain-adapted features so that we can influence, over time, the pedestrian Re-ID task. We call this CNN the *ReID network* R, trained with cross-entropy loss for standard classification by feeding it the output feature maps of the source images encoded by G_{enc}. This can be expressed as:

$$\mathcal{L}_{ReID}(G_{enc}, R, \boldsymbol{X}) = -\mathbb{E}_{\mathbf{x},\mathbf{q}\sim\boldsymbol{X}}\left[\sum_{i=1}^{P} \mathbf{q}_i \log\left(\frac{\exp(R_i(G_{enc}(\mathbf{x})))}{\sum_{j=1}^{P}\exp(R_j(G_{enc}(\mathbf{x})))}\right)\right] \tag{5.10}$$

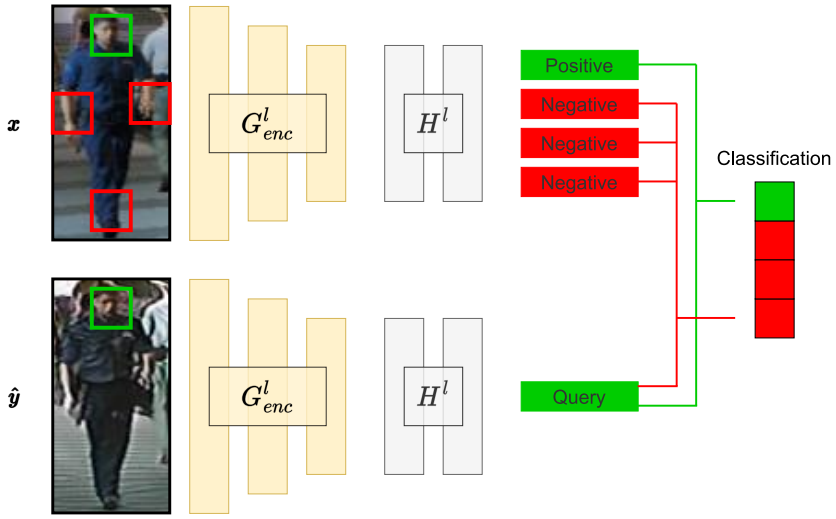

Figure 5.3: Classifying image patches, inspired from [6]. Given a generated image $\hat{\mathbf{y}}$, we first need to encode the query region $\hat{\mathbf{z}}^s_l$ (green). Then, for the source image $\mathbf{x}$, we have to encode both the positive region, $\mathbf{z}^s_l$ (green), as well as the negative ones, $\mathbf{z}^{S/s}_l$ (red). It is now possible to compute the cross-entropy loss for this $N+1$ classification problem and finally iterate over different queries and G_{enc} layers.

where P is the total number of pedestrians in the source dataset X and $\mathbf{q}_i$ is the i^{th} entry of $\mathbf{q}$, the one-hot encoded ground-truth label vector. Notice that, since the source labels are available, in this case we are sampling a datapoint from $\boldsymbol{X}$ consisting of a tuple having both an image and its label from which we can construct the one-hot encoded vector $\mathbf{q}$. The expression $R_k(G_{enc}(\mathbf{x}))$ means that we first feed the generator's encoder with an image $\mathbf{x}$. Then, the resulting feature map is passed through the $ReID$ network which outputs unnormalized scores (logits) for each of the P classes. The index k represents the k^{th} of those scores. However, as [61] points out, such loss could make the model overfit to the training identities. Since the identities of the test set are never seen during training (see chapter 3), we could encourage the model to be less confident on the training pedestrians and achieve better generalization capabilities. Work [41] proposes *label smoothing* as a solution. In eq. 5.10, each entry of the one-hot label encoded vector $\mathbf{q}$ associated to an image $\mathbf{x}$ is defined as follow:

$$\mathbf{q}_i = \begin{cases} 1 & \text{if } y = i, \\ 0 & \text{if } y \neq i \end{cases} \tag{5.11}$$

where y is the label of the person depicted in $\mathbf{x}$ and $i \in \{1, 2, ..., P\}$. With label smoothing, we soften the hard predictions decreasing the confidence of the largest logit and allowing information from the other ones. This regularization mechanism

can be adopted in the cross-entropy loss by simply changing eq. 5.11 into:

$$\mathbf{q}_i = \begin{cases} (1-\epsilon) & \text{if } y = i, \\ \epsilon / P & \text{if } y \neq i \end{cases} \tag{5.12}$$

where P is the total number of identities in the training set and ϵ is a small constant (0.1). The main rationale behind this model is that we want to exploit translated features that represent some coarse information from the source images. These features come from the last layer of G_{enc} and are responsible for the translation. They do not hold any information about the pedestrians' identity, but they still encode some low-level information comparable to the initial layers of an usual CNN. By feeding those features to the $ReID$ network, we can specialize them to be representative of the identities while maintaining target-domain information (see figure 5.4). This is done to prevent learning from the output images and save in computations. We employ a Resnet50 [4] starting from the second convolutional block and we add a batch normalization layer after the global average pooling as in [61].

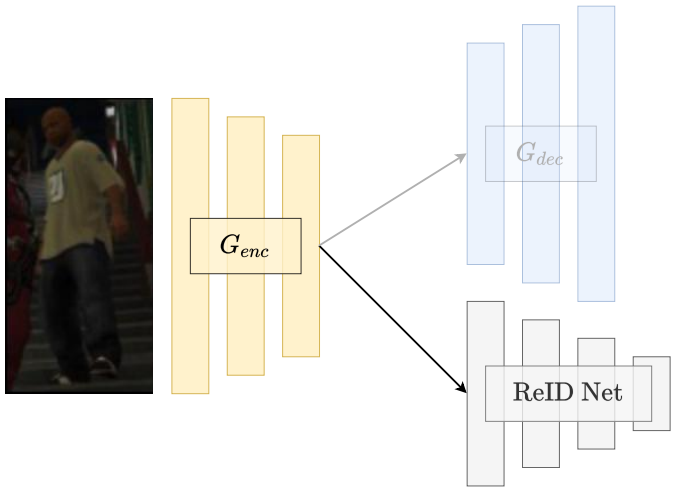

Figure 5.4: The Resnet-based network R receives the output of the G_{enc} and is optimized for classification on the training identities. For evaluation, we pass each target image $\mathbf{y}$ into $R(G_{enc}(\mathbf{y}))$. The (transparent) decoder G_{dec} is not directly involved in the Re-ID task.

5.1.4 Overall objective

Putting all together, we can finally state the overall objective:

$$\begin{aligned} L_{Total} = \mathcal{L}_{Gan}(G, D, \boldsymbol{X}, \boldsymbol{Y}) + w_R \mathcal{L}_{ReID}(G_{enc}, R, \boldsymbol{X}) + \\ + w_Y \mathcal{L}_{PatchNCE}(G, H, \boldsymbol{Y}) + w_X \mathcal{L}_{PatchNCE}(G, H, \boldsymbol{X}) \end{aligned} \tag{5.13}$$

where $\mathcal{L}_{Gan}(G, D, \boldsymbol{X}, \boldsymbol{Y})$ and its respective weights are defined in eq. 5.6, w_X and w_Y are the weights for the PatchNCE and identity PatchNCE losses, and w_R is the weight for the classification loss. When evaluating on target dataset test set, we pass the images through $R(G_{enc})$ to extract features for the ranking. These features should hopefully be strong pedestrian descriptors and hold information about the target data so that they can generalize better than a baseline trained only on the source dataset. As we explained above, this was done by feeding translated features to the classification network.

5.2 Experiments

In this section we will first provide all the training and evaluation details, underlying the challenges that arise when dealing with such kinds of models. Then, we will present a brief ablation analysis and reveal our results. We expand on this by providing qualitative results and explaining some of the incompatibilities that we encountered between discriminative learning and image translation.

5.2.1 Training details

Although CUT [6] is simpler than models such as CycleGans [7], it is still a complex architecture, even more so when coupled with discriminative feature learning. There are multiple optimizers, schedulers and losses to integrate together, rendering the hyperparameter search an arduous task. There has been a lot of trial and error for finding suitable architectures and hyperparameters, early stopping unsuccessful runs and repeating three times relevant experiments with different random seeds. A successful model should be able to balance the generator and discriminator losses while, at the same time, learning pedestrian descriptors. For the training process, we exploit our whole labeled source synthetic dataset GTASynthReid and the unlabeled training partition of a real target dataset. We perform source-to-target translation jointly with pedestrian classification on the source identities. These identities are encoded by feature vectors containing target-related information. We compare the results of our approach to a ResNet50 baseline trained only on our synthetic dataset for direct transfer. The baseline was trained with ADAM optimizer and cross-entropy for 60 epochs and a batch size of 32. Its initial learning rate is equal to 2e−4, while the weight decay is 5e−4. We multiplied the learning rate by 0.1 after the 20^{th}, 30^{th} and 40^th epochs.

For the generator instead, we employed a ResNet-like structure with 9 residual blocks as in CUT and instance normalization [23]. We also implemented the same discriminator, PatchGan [47], as explained in subsection 5.1.1. For the discriminator's feature matching loss, we compare the features of all the first 4 convolutional blocks. Regarding the feature-specialized networks, as in CUT, each

MLP samples at most 256 spatial locations (that are embeddings of the input image patches). Considering the generator's encoder only, these spatial locations are sampled from the first layer (reflective padding), the first and second convolutional blocks, the first and fifth residual blocks. The last component of this architecture is the $ReID$ network, trained for classification. Its structure is the same as a ResNet50, but we remove everything up to the second convolutional block. The features coming from the last layer of G_{enc} are directly passed through the $ReID$ net (which is as if we were passing them to the second convolutional layer of a ResNet50). We tried a different version of this by building a CNN with fewer layers but more channels per feature map. Through experiments, we noticed that the ResNet-based $ReID$ network achieved better results on the target datasets, despite having way fewer parameters.

For the optimizers, we tried SGD, but ADAM performed better as in [6, 7]. We trained for 60 epochs with a batch size of 8. All the learning rates have initial value of 2e−4. Those of the MLPs, generator and discriminator are multiplied by 0.1 after the 40^{th} epoch, while that of the $ReID$ net is multiplied by the same factor after the 30^{th} and 40^{th} epochs. The weight decay relative to the MLPs has been fixed to 5e−3, while the remaining ones are set to 5e−4. For each target dataset (Market [1], Duke [2], CUHK03 [3]) we used the available (unlabeled) training data for the translation. As source data we adopted our GTASynthReid dataset where, for each pedestrian we sampled at most 20 bouding boxes, using 10,760 images out of the total 94,312 ones. The source and target images are resized to 256×128. We applied random horizontal flip and random crop transforms to the training frames. Regarding the loss weights, we fix $w_D = 0.5$, $w_G = 1$, $w_{SD} = 0.5$, $w_{SG} = 0.5$, $w_R = 0.5$, $w_{\boldsymbol{Y}} = 0$ and $w_{\boldsymbol{X}} = 0.5$. For the identity mapping associated to $\mathcal{L}_{PatchNCE}$ we found better qualitative results on the image translation when $w_{\boldsymbol{Y}} = 0.5$, although the best person Re-ID results appeared when $w_{\boldsymbol{Y}} = 0$. We will develop more this point in the results section.

5.2.2 Evaluation details

We follow the standard evaluation protocols [56] of person Re-ID and cross-domain person Re-ID, similarly to the works presented in the previous chapter. Given a model trained from our source to a real dataset, we evaluate that model by extracting feature vectors with $R(G_{enc})$ and performing the retrieval on the target test partition. We underline that the identities of the target test set have never been seen during training. We can think of this as creating a model which adapts from a synthetic to a specific real setting. To relax this assumption and test a broader synth-to-real adaptation, we test each model also on the target test pedestrians of the remaining two datasets. We use CMC and MaP to measure the quality of the ranking and compare it to the baseline. We applied horizontal feature flipping to

the testing images and performed re-ranking [56].

Regarding the validation step, for this task it is often overlooked. In fact, the works that we analyzed skipped it. The reasons might vary. As we explained in chapter 3, there is not an unified validation approach for person Re-ID. Considering the more general domain adaptation field instead, validation is an active topic of research, being adopted and executed in several manners depending on the context. In the absence of reference points, we tried to develop our approach for validation during translation. For each source-to-target experiment, we extracted 200 identities from the training target set and used them as a validation partition. In this scenario, we would have to tune the hyperparameters search on the validation data, retraining later the best performing model with all the training data. However, we find out that this procedure is neither representative of target test error, nor useful for finding good hyperparameters. Consequently, we dropped this approach and followed the usual aforementioned pipeline.

5.2.3 Ablation studies

We can now see how each component of the total loss (eq. 5.13) affects the person Re-ID performance. In table 5.1, *Ours* stands for our model coupled with the generator, discriminator, cross-entropy and PatchNCE losses. We then add the feature matching loss $\mathcal{L}_{Sim}$ (eq. 5.5, 5.4) and the identity mapping PatchNCE written as $\mathcal{L}_{Idt}$ (5.9) for simplicity. When we append $\mathcal{R}$, it means that we used re-ranking. It can be seen that, relative to the pedestrian Re-ID, the base model coupled with the feature matching loss ($\mathcal{L}_{Sim}$) performs slightly better both with and without re-ranking. However, this only refers to the retrieval performance, without considering the generated image quality. We will further elaborate on this topic in the next subsections.

From now on, *Ours* refers to the best performing model loss configuration (as we just explained it). We compare it to the Resnet50 baseline (direct transfer) on three real-world datasets: Market [1], Duke [2], CUHK03 [3]. For each of those, instead of just evaluating the GTASynthReid-to-target adaptation, we also test the translation implemented on the other two real datasets. We try with and without re-ranking and repeat each experiment with three different random seeds. We noticed that the trials without re-ranking perform better on higher ranks of CMC. However, usually MaP and the first rank of CMC are given more importance. Thus, we consequently consider those two for the model selection. For Market (table 5.2) and CUHK03 (table 5.4) we find out the best models to respectively be GTASynthReid-to-Market and GTASynthReid-to-CHUK03, with re-ranking. For Duke instead (table 5.3), GTASynthReid-to-Market performed slightly better than GTASynthReid-to-Duke. We can now compare these results with those of similar existing approaches.

Model	Market				
	MaP	R1	R5	R10	R20
Ours	26.5	55.7	73.1	79.4	85.0
Ours$_{\mathcal{R}}$	40.9	59.8	71.5	76.6	81.6
Ours+$\mathcal{L}_{Sim}$	27.7	56.8	**73.7**	**80.5**	**85.7**
Ours$_{\mathcal{R}}$+$\mathcal{L}_{Sim}$	**42.6**	**61.2**	72.3	77.7	82.2
Ours+$\mathcal{L}_{Idt}$	26.6	55.8	73.2	79.8	85.0
Ours$_{\mathcal{R}}$+$\mathcal{L}_{Idt}$	41.3	59.8	71.0	76.0	81.1
Ours+$\mathcal{L}_{Sim}$+$\mathcal{L}_{Idt}$	26.3	55.5	72.6	78.9	84.7
Ours$_{\mathcal{R}}$+$\mathcal{L}_{Sim}$+$\mathcal{L}_{Idt}$	40.8	59.5	70.5	75.5	80.6

Table 5.1: Contribution of the feature matching loss ($\mathcal{L}_{Sim}$) and the identity mapping loss ($\mathcal{L}_{Idt}$) on top of the base model (in this table *Ours* includes the generator, discriminator, cross-entropy and PatchNCE losses). We evaluate on Market with and without re-ranking [56] (written as $\mathcal{R}$). Best results are in bold.

Model	**Target**	Market				
		MaP	R1	R5	R10	R20
Baseline	-	18.0	38.4	56.4	64.0	71.2
Baseline$_{\mathcal{R}}$	-	23.3	39.4	53.2	59.5	65.8
Ours	Market	27.7	56.8	**73.7**	**80.5**	**85.7**
Ours$_{\mathcal{R}}$	Market	**42.6**	**61.2**	72.3	77.7	82.2
Ours	Duke	26.2	54.8	72.5	79.2	85.0
Ours$_{\mathcal{R}}$	Duke	40.7	59.5	70.9	75.9	80.7
Ours	CUHK03	26.8	55.6	72.9	79.4	85.2
Ours$_{\mathcal{R}}$	CUHK03	41.6	60.0	71.4	76.3	81.3

Table 5.2: Comparison between the baseline (Resnet50 trained on GTASynthReid for direct transfer) and the best GTASynthReid-to-target performing models evaluated on the Market dataset, both with and without re-ranking [56] (written as $\mathcal{R}$). Best results are in bold.

5.2.4 Results

In this section, we compare our framework to the state of the art. Since we employed CUT, we compare with other generative real-to-real approaches and general synth-to-real methods that appeared in top conferences. For a more extensive performance overview, in the previous three chapters, we showed the results of supervised Re-ID methods and broader domain adaptive frameworks. As one can see from tables 5.5, 5.6 and 5.7, we addressed works that implement both real-to-real and synth-to-real

Model	**Target**	Duke				
		MaP	R1	R5	R10	R20
Baseline	-	12.7	24.3	39.2	46.0	53.1
Baseline$_{\mathcal{R}}$	-	18.9	27.7	39.2	45.4	52.1
Ours	Market	20.1	39.2	54.6	61.1	**66.8**
Ours$_{\mathcal{R}}$	Market	**32.8**	**44.8**	**55.9**	**61.2**	66.0
Ours	Duke	19.7	38.2	53.6	59.8	66.1
Ours$_{\mathcal{R}}$	Duke	31.8	43.2	54.0	59.6	65.1
Ours	CUHK03	19.3	38.2	53.4	59.6	65.9
Ours$_{\mathcal{R}}$	CUHK03	31.9	43.6	55.0	60.5	65.6

Table 5.3: Comparison between the baseline (Resnet50 trained on GTASynthReid for direct transfer) and the best GTASynthReid-to-target performing models evaluated on the Duke dataset, both with and without re-ranking [56] (written as $\mathcal{R}$). Best results are in bold.

Model	**Target**	CUHK03				
		MaP	R1	R5	R10	R20
Baseline	-	5.2	5.3	11.4	15.7	21.2
Baseline$_{\mathcal{R}}$	-	6.7	5.6	11.2	15.1	19.5
Ours	Market	4.9	4.9	11.3	15.9	21.5
Ours$_{\mathcal{R}}$	Market	7.6	6.9	11.4	14.8	19.8
Ours	Duke	4.8	4.9	11.1	15.6	21.5
Ours$_{\mathcal{R}}$	Duke	7.2	6.2	11.1	14.4	18.8
Ours	CUHK03	5.3	5.4	11.8	**16.5**	**22.9**
Ours$_{\mathcal{R}}$	CUHK03	**8.3**	**7.2**	**12.2**	16.0	21.3

Table 5.4: Comparison between the baseline (Resnet50 trained on GTASynthReid for direct transfer) and the best GTASynthReid-to-target peforming models evaluated on the CUHK03 dataset, both with and without re-ranking [56] (written as $\mathcal{R}$). Best results are in bold.

adaptation. Although being able to generalize from a real to another real dataset might be easier rather than starting from computer-generated data, it is crucial to see how far works that make use of synthetic data have come, by means of performance. This allows us to understand whether such a novel method would be applicable in a real-world scenario. For each real dataset, we show the scores of the best performing model with re-ranking [56]. For the Market [1] dataset, we compare with our GTASynthReid-to-Market model (tables 5.5, 5.2). For Duke [2] instead, we compare again with our GTASynthReid-to-Market model since it performed

better than GTASynthReid-to-Duke (tables 5.6, 5.3). Finally, for CUHK03 [3] we compare with our GTASynthReid-to-CUHK03 model (tables 5.7, 5.4).

Model	**Venue**	**Source**	Market	
			MaP	R1
Real data				
SPGAN[86]	CVPR18	Duke	26.9	58.1
Cyclegan[78]	CVPR19	Duke	24.5	52.0
ISSDA[78]	CVPR19	Duke	63.1	81.3
CR-GAN[87]	IICV19	Duke	33.2	64.5
CR-GAN[87]	IICV19	CUHK03	30.4	58.5
CR-GAN+TAUDL[87]	IICV19	Duke	54.0	77.7
CR-GAN+TAUDL[87]	IICV19	CUHK03	56.0	78.3
DG-Net++[79]	EECV20	Duke	61.7	82.1
DG-Net++[79]	EECV20	MSMT17	64.6	83.1
GLC[80]	CVPR21	None	66.8	87.3
GLC[80]	CVPR21	Duke	**75.4**	**90.5**
Synthetic data				
SyRI[91]	EECV18	SyRI	-	65.7
Resnet50[95][4]	ACMM20	RandPerson	28.8	55.6
CBN[89][64]	CVPR21	UnrealPerson	54.3	79.0
BoT[89][61]	CVPR21	UnrealPerson	37.2	64.0
JVTC[89][88]	CVPR21	UnrealPerson	**80.2**	**93.0**
Ours	Thesis	GTASynthReid	42.6	61.2
BoT [61]	Thesis	GTASynthReid	44.7	64.6

Table 5.5: We compare our method to generative real-to-real adaptation works and general synth-to-real approaches, when evaluated on Market. We both show the scores of our best model and the performance of BoT [61] when trained on GTASynthReid. Best results are in bold, both for real-to-real and synt-to-synth works.

As we can see from these tables, our join discriminative and domain transfer method is able to surpass (at least in one metric) some earlier real-to-real generative approaches for Market ([86, 78, 87]) and Duke ([86, 78, 87]). In the synth-to-Market adaptation setting we can surpass [95] and [89] in MaP when it uses BoT [61]. Regarding the synth-to-Duke setting instead, we can surpass [95] in MaP. For CUHK03, we reach a comparable MaP to [95]. Although this last dataset is interesting, we noticed that it is often not employed as a target dataset for adaptation methods. It seems to be more arduous to achieve a good performance for this dataset in our scenario. Comparing these newer frameworks, we notice the

Model	**Venue**	**Source**	Duke	
			MaP	R1
Real data				
SPGAN[86]	CVPR18	Market	26.4	46.9
Cyclegan[78]	CVPR19	Market	19.4	35.7
ISSDA[78]	CVPR19	Market	54.1	72.8
CR-GAN[87]	IICV19	Market	33.3	56.0
CR-GAN[87]	IICV19	CUHK03	26.9	46.5
CR-GAN+TAUDL[87]	IICV19	Market	48.6	68.9
CR-GAN+TAUDL[87]	IICV19	CUHK03	47.7	67.7
DG-Net++[79]	EECV20	Market	63.8	78.9
DG-Net++[79]	EECV20	MSMT17	58.2	75.2
GLC[80]	CVPR21	None	62.8	**82.9**
GLC[80]	CVPR21	Market	**67.6**	81.9
Synthetic data				
Resnet50[95][4]	ACMM20	RandPerson	27.1	47.6
CBN[89][64]	CVPR21	UnrealPerson	49.4	69.7
BoT[89][25]	CVPR21	UnrealPerson	37.5	58.0
JVTC[89][88]	CVPR21	UnrealPerson	**75.2**	**88.3**
Ours	Thesis	GTASynthReid	32.8	44.8
BoT[61]	Thesis	GTASynthReid	41.9	53.3

Table 5.6: We compare our method to generative real-to-real adaptation works and general synth-to-real approaches, when evaluated on Duke. We both show the scores of our best model and the performance of BoT [61] when trained on GTASynthReid. Best results are in bold, both for real-to-real and synt-to-synth works.

tendency to work more on the dataset, rather than on the modeling. Works such as [95, 89] created huge image collections depicting far more identities (respectively 8,000 and 3,000) that we did (only 538). As explained in the previous chapter, they then employed already existing models, which are lighter than ours and thus can sample more images during training (we only sample 10,760 images). Furthermore, when work [89] adopts [88] for unsupervised domain adaptation from its dataset, it can even beat state of the art [80, 79] real-to-real methods. This is an important future research direction since operating with computer-generated data could lift some of the real data burdens and make up for the variations that we would not be able to record in a real-world scenario. However, we point out that our work was bounded in using GTAV to generate a synthetic dataset. In this game, we only found 538 usable pedestrians. We tried adding more of them, but they had poor

Model	Venue	Source	CUHK03	
			MaP	R1
Synthetic data				
Resnet50[95][4]	ACMM20	RandPerson	10.8	13.4
Ours	Thesis	GTASynthReid	8.3	7.2
BoT[61]	Thesis	GTASynthReid	9.6	8.5

Table 5.7: We compare our method to generative real-to-real adaptation works and general synth-to-real approaches, when evaluated on CUHK03. We both show the scores of our best model and the performance of BoT [61] when trained on GTASynthReid. Best results are in bold, both for real-to-real and synt-to-synth works.

body annotations that did not allow us to generate automatic bounding boxes. Having said that, we tried BoT on our dataset as [89] did on theirs. With re-ranking, we were able to achieve better performance than with our joint discriminative domain mapping method. One might ask how a direct transfer method can surpass a domain adaptive one. We think that this is due to the triplet loss in BoT which can help in building features that can better recognize whether two characters share the same identity (pushing together pedestrians with the same identity and pulling apart those with different ones). Another reason could lie in the incompatibility between feature learning and domain transfer. We assumed that translating images from one domain to another could help a CNN to learn transferred discriminant features. In the next subsection, we will expand more on these nuances.

5.2.5 Qualitative results

The aim of this work lies in adapting our synthetic dataset to real ones, by translating the computer-generated images. Ideally, we would both have strong pedestrian descriptors for the target dataset and generated images that look more realistic. However, as we just said, it is not a given that generating more realistic images from a human perspective combines well with feature discriminative learning. Indeed, we noticed that the configuration giving the best retrieval scores does not occur to return the best images (see figure 5.5). The previous results were obtained without considering the identity mapping PatchNCE loss ($w_{\mathcal{Y}} = 0$). In this scenario, the output images had some artifacts and unrealistic color backgrounds. However, when including the identity mapping loss ($w_{\mathcal{Y}} = 0.5$) and discarding the feature matching one ($w_{\mathcal{SD}} = w_{\mathcal{SG}} = 0$), their quality seems to improve (from a human perception point of view). Unfortunately, as we saw in subsection 5.2.3, adding this term in the total loss decreases the performance of the person Re-ID. Nevertheless,

we believe that, for the sole purpose of generating better images, removing the feature matching loss and adding the identity mapping one overcomes the unrealistic colors and better guides the image generation [6]. Even if artifacts are still present (especially for the more difficult CUHK03 dataset), they are less frequent compared to the other configuration. There are many ways to evaluate this. One could ask a group of people to recognize whether the generated images are real or fake with online services such as *Amazon Mechanical Turk* [7]. Another approach is to use metrics that compare the features of real and generated images from a deep model. To this extent, we adopt the *Fréchet Inception Distance* [9] (FID) which employs the Inception [41] architecture as its name suggests (the lower the better). Appendix B provides more details. We first compute this metric between our dataset and the three other real ones. Then, for each of them, we translate GTASynthReid and finally compare GTASynthReid-to-Market with Market, GTASynthReid-to-Duke with Duke and GTASynthReid-to-CUHK03 with CUHK03. We use the configuration where $w_{\mathcal{Y}} = 0.5$ and $w_{\mathcal{SD}} = w_{\mathcal{SG}} = 0$ over three different random seeds. From table 5.8, one can notice how the translation improves the FID relative to measuring that distance between our original GTASynthReid and the real data. In particular, for Market and CUHK03, we were able to decrease the FID by approximately 50%, while for Duke by 60%. We conclude this section by showing in figure 5.6 the translation of some synthetic pedestrians to the three real domains.

Translation	**Target**	**FID**
GTASynthReid	Market	45.14
GTASynthReid-to-Market	Market	**24.29**
GTASynthReid	Duke	50.44
GTASynthReid-to-Duke	Duke	**19.53**
GTASynthReid	CUHK03	63.23
GTASynthReid-to-CUHK03	CUHK03	**31.54**

Table 5.8: For each real dataset, we first compute the FID between our GTASynthReid and the real dataset itself. Then we also compute the FID between the tranlated GTASynthReid-to-real and the real data. We can see that the translation successfully brings to smaller distances (the smaller the better). Best result in bold relative to each real dataset.

Figure 5.5: The pictures of each column have been translated from our synthetic dataset to respectively the Market, Duke and CUHK03 targets. Each column has two rows. In the first one, we used the best performing configuration, while in the second one we put $w_{\mathcal{Y}} = 0.5$ and $w_{\mathcal{SD}} = w_{\mathcal{SG}} = 0$. One can notice that the images in the first row present artefacts, odd colors and unexpected shadows.

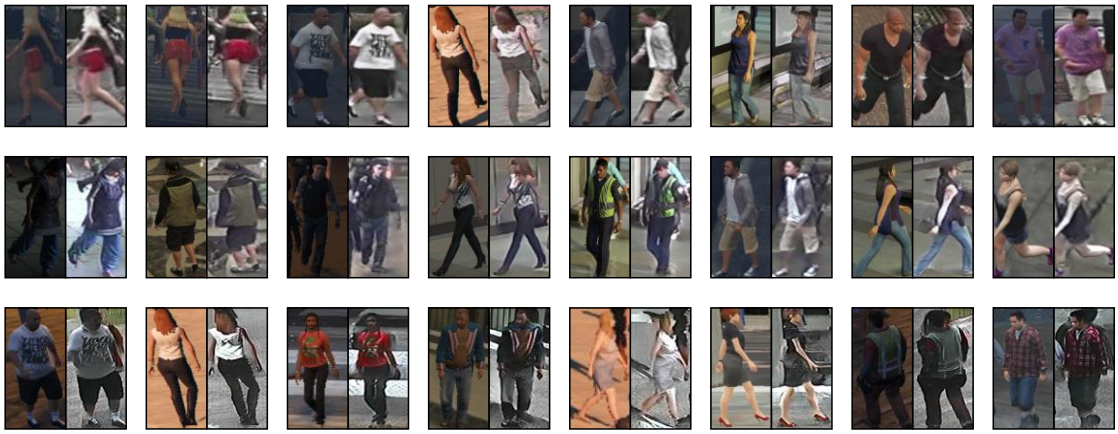

Figure 5.6: Transferred pedestrians. For each of them, we show the original (left) and translated (right) images from our GTASynthReid to Market, Duke and CUHK03 (first, second and third rows).

Chapter 6

Conclusions

Before ending this thesis, we provide some final broad insights about our work compared to similar ones. Moreover, we also try to envision the direction this field might take in the future.

6.1 About this work

Starting from our synthetic dataset for pedestrian Re-ID, our objective was to generalize well across different real domains. There are several ways one could approach this problem, we chose to investigate generative methods. To the best of our knowledge, we are the first to apply the generative contrastive framework for unpaired translation CUT [6] to person Re-ID. Furthermore, we integrated it with a discriminative module to inject target domain information into the network we employ for retrieval. Then, we showed that our architecture adapts better to the target data than a standard Resnet50 [4] baseline trained for direct transfer. Despite many works evaluating their methods only on the target dataset, to better uncover generalization capabilities, we also tested our framework on the remaining real datasets. Specifically, we were able to approximately maintain a similar performance on Market and Duke, surpassing the GTASynthReid-to-Duke result when training on Market and CUHK03. While we did not overtake the most recent generative real-to-real and broad synth-to-real transfer works, except for [95], we achieved better performances than earlier methods that employed, for instance, CycleGans. Recent general synth-to-real approaches rely on large datasets with far more identities, while we were limited by the allowed pedestrians of GTAV. Generative real-to-real frameworks depend instead on bigger architectures, having a couple of encoders in the generator and often adopting some kind of cycle consistency loss [79, 80]. Although CUT is not simple, it is easier to train than those techniques that encode, beyond appearance, also pose information. It does

not require a double network structure and, instead of relying on cycle consistency, it preserves the input semantics by comparing corresponding input-output patches. The adoption of this method for person Re-ID is the very heart of our work. Paired with our similarity loss and joint discriminative learning, this thesis is a simpler yet effective one-stage generative framework for synth-to-real person Re-ID. We also point out that, for hardware limitations, we often had to sample less data than existing works, especially compared to recent synth-to-real models that use far more pedestrians. On a qualitative level, we show that our procedure generates images with style more similar to the real target dataset. In this sense, we tried to generate more realistic images through unpaired translation, which is something that transcends person Re-ID.

6.2 Future directions

One of the major drawbacks of our dataset is the low number of distinct pedestrians. For future improvements, one could try to design new characters to embed into the video game GTAV. Having more identities seems to be the predominant direction of similar works. This will render the synth-to-real Re-ID problem more arduous and possibly deliver better results on the target data. Recent generative works disentangle appearance from other inherent information by rendering the same pedestrian to match, for instance, different poses across domains. A natural extension of our work would be to devise a method that performs such disentanglement without cycle consistency beyond matching the target style. This could lead to smaller and easier to train models that perform better than methodologies working via standard translation. However, CUT asks corresponding input-output patches to be similar, which can be inconsistent with this kind of disentanglement unless we redefine the concept of patch similarity or design other losses. It is not an easy task and more research is needed.
On a more general level, while working on this task, we came across several cues we feel will become increasingly more relevant in this field. First of all, there has been a paradigm shift in recent years leading to more research focused on domain transfer solutions for person Re-ID. Training and testing on the same data has less practical applications, since generalizing to new domains is fundamental for real-world Re-ID systems. However, collecting data for this task is arduous and time-consuming. Moreover, real-world datasets have growing ethical concerns about the lack of informed consent for being recorded. With these motivations, researchers are creating and adopting synthetic datasets also in this field. The general trend is to build large-scale datasets, with far more identities than their real counterparts. As we showed before, when coupled with powerful models, one can even surpass real-to-real adaptation when using computer-generated pedestrians.

We feel that, in the future, relevant research might focus more on building superior synthetic data rather than just improving existing models to perform better on benchmark datasets. Software-generated environments could also simplify more complex scenarios such as changing-clothes person Re-ID, being able to control the data-generating process. To this end, we think that working with synthetic worlds will inspire new interesting works and possibly help to address existing challenges.

Appendix A

Re-Ranking

The quality of the ranking process is crucial in retrieval applications such as person Re-ID. In these scenarios, we have, if any, different classes in the training and test partitions. After the feature extractions, we encode all the query and gallery images so that, for each probe image in the query, we rank the gallery set by ascending distance. The resulting ranking can be further improved *re-ranking* techniques. These will usually boost the rank accuracy, although they depend on the quality of the original ranking and, beyond rank-5, we noticed that they could worsen the results. We will briefly show how re-ranking can be applied to person Re-ID by following the notation of [56].

A.1 K-reciprocal encoding

Given a gallery set of N images $G = \{g_i \mid i = 1, 2, ..., N\}$ and a probe pedestrian p, we can define the distance between the feature embeddings of p and g_i as $d(p, g_i)$. The starting ranking, $\{g_1, g_2, ..., g_N\}$, is represented by the gallery pedestrians sorted by ascending distance from the probe. We can now define the top k most similar samples to p as $N(p, k) = \{g_1, g_2, ..., g_k\}$. Then, the *k-reciprocal nearest neighbors* of p are:

$$\mathcal{R}(p, k) = \{g_i \mid (g_i \in N(p, k)) \wedge (p \in N(g_i, k))\}. \tag{A.1}$$

This should return a list where more positive pedestrians are at the top. However, some of the positives might be excluded (there are less than k results) from the k-reciprocal nearest neighbors, due to illumination, pose and viewpoint variations [56]. This issue can be addressed by incrementally expanding $\mathcal{R}(p, k)$ with the $\frac{1}{2}k$-reciprocal nearest neighbors of each of its elements:

$$\begin{aligned} \mathcal{R}^*(p,k) &\leftarrow \mathcal{R}(p,k) \cup \mathcal{R}(q,\frac{1}{2}k) \\ \text{s.t.} \quad &\left|\mathcal{R}(p,k) \cap \mathcal{R}(q,\frac{1}{2}k)\right| \geq \frac{2}{3}\left|\mathcal{R}(q,\frac{1}{2}k)\right|, \ \forall q \in \mathcal{R}(p,k) \end{aligned} \tag{A.2}$$

where $\left|\cdot\right|$ counts the elements in a given set and s.t. denotes a constraint. After this process, $\mathcal{R}^*(p,k)$ could end up with more positive samples than $\mathcal{R}(p,k)$.

A.2 K-reciprocal distance

Given the probe p and a gallery image g_i, one can see how similar they are by computing the *Jaccard* distance of their k-reciprocal nearest neighbors, indicating their overlapping level. However, computing the intersection over union in this format is time-consuming and equally weighting all neighbors could lead to a not discriminative neighbor set [56]. They address the first issues by encoding the k-reciprocal nearest neighbors of p as a vector, $\mathcal{V}_p$, where each entry is 1 if the gallery image $g_i \in \mathcal{R}^*(p,k)$, with $i = 1, 2, ..., N$. To assign more weight to the neighbors closer to p, we can rewrite each entry of $\mathcal{V}p$ as:

$$\mathcal{V}_{p,g_i} = \begin{cases} e^{-d(p,g_i)} & \text{if } g_i \in \mathcal{R}^*(p.k) \\ 0 & \text{otherwise} \end{cases} \tag{A.3}$$

where $\mathcal{R}^*(p.k)$ is as explained in eq. A.2. We can now express the intersection and union between the k-reciprocal nearest neighbors of p and each g_i as:

$$\begin{aligned} \left|\mathcal{R}^*(p,k) \cap \mathcal{R}^*(g_i,k)\right| &= \|\min(\mathcal{V}_p, \mathcal{V}_{g_i})\|_1 \\ \left|\mathcal{R}^*(p,k) \cup \mathcal{R}^*(g_i,k)\right| &= \|\max(\mathcal{V}_p, \mathcal{V}_{g_i})\|_1 \end{aligned} \tag{A.4}$$

where $\|\cdot\|$ is the L1 norm. Having expressed everything in terms of vector operations, work [56] efficiently computes the Jaccard distance as follows:

$$d_J(p,g_i) = 1 - \frac{\sum_{j=1}^{N} \min(\mathcal{V}_{p,g_j}, \mathcal{V}_{g_i,g_j})}{\sum_{j=1}^{N} \max(\mathcal{V}_{p,g_j}, \mathcal{V}_{g_i,g_j})} \tag{A.5}$$

where $\mathcal{V}_{p,g_j}$ is defined as in eq. A.4. The final distance is computed as the combination of the original and Jaccard distances:

$$d^*(p,g_i) = (1-\lambda)d_J(p,g_i) + \lambda d(p,g_i) \tag{A.6}$$

where $\lambda \in [0,1]$ increases the resulting distance when g_i is far from p.

Appendix B

Evaluating Generative Adversarial Networks

Evaluating the image quality of GANs is a challenging yet interesting task. One can approach this problem in several ways. For instance, we could ask a group of people to label some pictures as real or fake to see how good our model is at generating realistic images. This is usually outsourced on online services such as *Amazon Mechanical Turk*, allowing to assign a large number of images to many workers, averaging the results. Another approach relies on using distance metrics. However, directly comparing image pixels does not provide representative results. We have no interest in obtaining generated pictures that look the same as the original ones on a pixel-by-pixel level. Instead, we care about the capability of generating fake images that look, on a perceptual level, indistinguishable from the real ones. To this end, we can exploit neural networks to extract features and work on those rather than the input images. We will briefly discuss the *Inception Score* (IS) [102] and the *Fréchet Inception Distance* (FID) [9], both of which employ an Inception network [41], usually pre-trained on ImageNet [33].

B.1 Inception Score

The IS combines two concepts that should manifest when generating new images [102]. Each output picture should contain meaningful objects, while the resulting images should collectively be varied enough. To test these two requirements, we have to feed an Inception network with the generated images and get the predictions. For the first one, we expect the conditional label distribution for each image to have a predominant class, *i.e.*, low entropy. For the second one, we ask the marginal cumulative label distribution over several images depicting different objects to be

close to the uniform, *i.e.*, high entropy. The IS is the exponential of the *Kullback-Leibler* (KL) divergence of these two distributions averaged over the output images:

$$\begin{aligned} KL(P(y \mid \mathbf{x}) \| P(y)) &= -\sum_{\mathbf{x} \in \boldsymbol{X}} P(y \mid \mathbf{x}) \log \left(\frac{P(y \mid \mathbf{x})}{P(y)} \right) \\ IS &= -\exp(KL(\mathbb{E}_{\mathbf{x} \in \boldsymbol{X}} P(y \mid \mathbf{x}) \| P(y))) \end{aligned} \tag{B.1}$$

where $P(y \mid \mathbf{x})$ is the conditional label distribution and $P(y)$ is the marginal cumulative label distribution. The more those distributions are dissimilar, the higher the score. In such a scenario, each image will depict a clear object and, at the same time, it will differ from the other pictures.

B.2 Fréchet Inception Distance

The main drawback of the IS is the lack of an actual comparison between real and fake images since real-world samples are not employed [9]. The FID addresses this by directly comparing the statistics of both sample spaces. For each real and fake image, we first have to capture meaningful features by extracting the activations of the last pooling layer of the Inception network. Then, we compute their mean and covariance for each set of images. Assuming that those activations follow a multivariate normal distribution [9], their difference is computed by the Fréchet distance (or *Wasserstein-2* distance). Since we are encoding each image with the Inception network, we can write the FID as:

$$d^2((\mathbf{m}, \boldsymbol{C}), (\mathbf{m}_r, \boldsymbol{C}_r)) = \|\mathbf{m} - \mathbf{m}_r\|_2^2 + \mathrm{Tr}(\boldsymbol{C} + \boldsymbol{C}_r - 2(\boldsymbol{C}\boldsymbol{C}_w)^{\frac{1}{2}}) \tag{B.2}$$

where $\mathbf{m}, \boldsymbol{C}$ are the mean and covariance of the generated images while $\mathbf{m}_r, \boldsymbol{C}_r$ are the mean and covariance of the real ones. In this case, lower values of FID are associated with higher output image quality. Work [9] shows that this metric can capture better similarities and analogies between real and generated images compared to the Inception score.

Bibliography

[1] Liang Zheng, Liyue Shen, Lu Tian, Shengjin Wang, Jingdong Wang, and Qi Tian. «Scalable Person Re-identification: A Benchmark». In: *Computer Vision, IEEE International Conference on*. 2015 (cit. on pp. 3, 33, 36, 37, 69, 77, 78, 80).

[2] 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*. 2017 (cit. on pp. 3, 37, 77, 78, 80).

[3] Wei Li, Rui Zhao, Tong Xiao, and Xiaogang Wang. «DeepReID: Deep Filter Pairing Neural Network for Person Re-identification». In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 152–159. DOI: 10.1109/CVPR.2014.27 (cit. on pp. 3, 33, 38, 77, 78, 81).

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. «Deep Residual Learning for Image Recognition». In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90 (cit. on pp. 3, 25, 40, 66, 67, 75, 81–83, 86).

[5] Kaiyang Zhou, Yongxin Yang, Andrea Cavallaro, and Tao Xiang. «Learning Generalisable Omni-Scale Representations for Person Re-Identification». In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021), pp. 1–1. DOI: 10.1109/TPAMI.2021.3069237 (cit. on pp. 3, 58).

[6] Taesung Park, Alexei A. Efros, Richard Zhang, and Jun-Yan Zhu. «Contrastive Learning for Unpaired Image-to-Image Translation». In: *European Conference on Computer Vision*. 2020 (cit. on pp. 3, 29, 30, 52, 53, 68, 69, 71–74, 76, 77, 84, 86).

[7] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. «Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks». In: *Computer Vision (ICCV), 2017 IEEE International Conference on*. 2017 (cit. on pp. 4, 29, 30, 52, 53, 73, 76, 77, 84).

[8] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. «Perceptual Losses for Real-Time Style Transfer and Super-Resolution». In: *CoRR* abs/1603.08155 (2016). arXiv: 1603.08155. URL: http://arxiv.org/abs/1603.08155 (cit. on pp. 4, 52, 70).

[9] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, Günter Klambauer, and Sepp Hochreiter. «GANs Trained by a Two Time-Scale Update Rule Converge to a Nash Equilibrium». In: *CoRR* abs/1706.08500 (2017). arXiv: 1706.08500. URL: http://arxiv.org/abs/1706.08500 (cit. on pp. 4, 84, 91, 92).

[10] John McCarthy, Marvin L. Minsky, Nathaniel Rochester, and Claude E. Shannon. «A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence, August 31, 1955». In: *AI Magazine* 27.4 (Dec. 2006), p. 12. DOI: 10.1609/aimag.v27i4.1904. URL: https://ojs.aaai.org/index.php/aimagazine/article/view/1904 (cit. on p. 6).

[11] Tom M. Mitchell. *Machine Learning*. New York: McGraw-Hill, 1997. ISBN: 978-0-07-042807-2 (cit. on p. 6).

[12] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. USA: Cambridge University Press, 2014. ISBN: 1107057132 (cit. on pp. 7, 8).

[13] Frank Rosenblatt. «The perceptron: a probabilistic model for information storage and organization in the brain.» In: *Psychological review* 65 6 (1958), pp. 386–408 (cit. on pp. 7, 8).

[14] D.G. Lowe. «Object recognition from local scale-invariant features». In: *Proceedings of the Seventh IEEE International Conference on Computer Vision*. Vol. 2. 1999, 1150–1157 vol.2. DOI: 10.1109/ICCV.1999.790410 (cit. on pp. 8, 39).

[15] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. «Speeded-Up Robust Features (SURF)». In: *Computer Vision and Image Understanding* 110.3 (2008). Similarity Matching in Computer Vision and Multimedia, pp. 346–359. ISSN: 1077-3142. DOI: https://doi.org/10.1016/j.cviu.2007.09.014. URL: https://www.sciencedirect.com/science/article/pii/S1077314207001555 (cit. on pp. 8, 39).

[16] L. Jing and Y. Tian. «Self-Supervised Visual Feature Learning With Deep Neural Networks: A Survey». In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43.11 (Nov. 2021), pp. 4037–4058. ISSN: 1939-3539. DOI: 10.1109/TPAMI.2020.2992393 (cit. on pp. 9, 51).

[17] Gareth M. James, Daniela M. Witten, Trevor J. Hastie, and Robert Tibshirani. «An introduction to statistical learning». In: 2021 (cit. on p. 10).

[18] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016 (cit. on pp. 10, 14, 19–21).

[19] Vinod Nair and Geoffrey E. Hinton. «Rectified Linear Units Improve Restricted Boltzmann Machines». In: *Proceedings of the 27th International Conference on International Conference on Machine Learning*. ICML'10. Haifa, Israel: Omnipress, 2010, pp. 807–814. ISBN: 9781605589077 (cit. on p. 13).

[20] Min Lin, Qiang Chen, and Shuicheng Yan. «Network In Network». In: *CoRR* abs/1312.4400 (2014) (cit. on pp. 14, 26).

[21] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. «Dropout: A Simple Way to Prevent Neural Networks from Overfitting». In: *Journal of Machine Learning Research* 15.56 (2014), pp. 1929–1958. URL: http://jmlr.org/papers/v15/srivastava14a.html (cit. on p. 14).

[22] Sergey Ioffe and Christian Szegedy. «Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift». In: *CoRR* abs/1502.03167 (2015). arXiv: 1502.03167. URL: http://arxiv.org/abs/1502.03167 (cit. on pp. 14, 15, 40).

[23] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. *Instance Normalization: The Missing Ingredient for Fast Stylization*. 2017. arXiv: 1607.08022 [cs.CV] (cit. on pp. 15, 76).

[24] Xingang Pan, Ping Luo, Jianping Shi, and Xiaoou Tang. «Two at Once: Enhancing Learning and Generalization Capacities via IBN-Net». In: *CoRR* abs/1807.09441 (2018). arXiv: 1807.09441. URL: http://arxiv.org/abs/1807.09441 (cit. on p. 16).

[25] Alexander Hermans, Lucas Beyer, and Bastian Leibe. «In Defense of the Triplet Loss for Person Re-Identification». In: *CoRR* abs/1703.07737 (2017). arXiv: 1703.07737. URL: http://arxiv.org/abs/1703.07737 (cit. on pp. 17, 18, 40, 67, 82).

[26] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. «Signature Verification using a "Siamese" Time Delay Neural Network». In: *Advances in Neural Information Processing Systems*. Ed. by J. Cowan, G. Tesauro, and J. Alspector. Vol. 6. Morgan-Kaufmann, 1994. URL: https://proceedings.neurips.cc/paper/1993/file/288cc0ff022877bd3df94bc9360b9c5d-Paper.pdf (cit. on pp. 17, 39, 42).

[27] R. Hadsell, S. Chopra, and Y. LeCun. «Dimensionality Reduction by Learning an Invariant Mapping». In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*. Vol. 2. 2006, pp. 1735–1742. DOI: 10.1109/CVPR.2006.100 (cit. on p. 17).

[28] Florian Schroff, Dmitry Kalenichenko, and James Philbin. «FaceNet: A unified embedding for face recognition and clustering». In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 815–823. DOI: 10.1109/CVPR.2015.7298682 (cit. on p. 17).

[29] Diederik Kingma and Jimmy Ba. «Adam: A Method for Stochastic Optimization». In: *International Conference on Learning Representations* (Dec. 2014) (cit. on pp. 21, 22).

[30] Wouter M. Kouw. «An introduction to domain adaptation and transfer learning». In: *CoRR* abs/1812.11806 (2018). arXiv: 1812.11806. URL: http://arxiv.org/abs/1812.11806 (cit. on pp. 22, 47–49).

[31] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. «Gradient-based learning applied to document recognition». In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324. DOI: 10.1109/5.726791 (cit. on p. 24).

[32] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. «ImageNet Classification with Deep Convolutional Neural Networks». In: *Advances in Neural Information Processing Systems*. Ed. by F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger. Vol. 25. Curran Associates, Inc., 2012. URL: https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf (cit. on p. 24).

[33] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. «ImageNet: A large-scale hierarchical image database». In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848 (cit. on pp. 24, 40, 91).

[34] Karen Simonyan and Andrew Zisserman. «Very Deep Convolutional Networks for Large-Scale Image Recognition». In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2015. URL: http://arxiv.org/abs/1409.1556 (cit. on pp. 24, 52).

[35] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In-So Kweon. «CBAM: Convolutional Block Attention Module». In: *ECCV*. 2018 (cit. on pp. 25, 55).

[36] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. «Non-local Neural Networks». In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, pp. 7794–7803. DOI: 10.1109/CVPR.2018.00813 (cit. on pp. 25, 55).

[37] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. «Attention is All you Need». In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf (cit. on p. 25).

[38] Sepp Hochreiter and Jürgen Schmidhuber. «Long Short-Term Memory». In: *Neural Comput.* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. URL: https://doi.org/10.1162/neco.1997.9.8.1735 (cit. on p. 25).

[39] Mingxing Tan and Quoc Le. «EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks». In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, Sept. 2019, pp. 6105–6114. URL: https://proceedings.mlr.press/v97/tan19a.html (cit. on p. 25).

[40] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. «Going Deeper with Convolutions». In: *CoRR* abs/1409.4842 (2014). arXiv: 1409.4842. URL: http://arxiv.org/abs/1409.4842 (cit. on p. 26).

[41] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. «Rethinking the Inception Architecture for Computer Vision». In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), pp. 2818–2826 (cit. on pp. 26, 65, 74, 84, 91).

[42] Ian J. Goodfellow. «NIPS 2016 Tutorial: Generative Adversarial Networks». In: *CoRR* abs/1701.00160 (2017). arXiv: 1701.00160. URL: http://arxiv.org/abs/1701.00160 (cit. on pp. 27–29).

[43] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. «Generative Adversarial Nets». In: *Advances in Neural Information Processing Systems*. Ed. by Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger. Vol. 27. Curran Associates, Inc., 2014. URL: https://proceedings.neurips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf (cit. on pp. 26, 28, 29).

[44] Alec Radford, Luke Metz, and Soumith Chintala. «Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks». In: *CoRR* abs/1511.06434 (2016) (cit. on pp. 26, 27, 30).

[45] Martin Arjovsky, Soumith Chintala, and Léon Bottou. «Wasserstein Generative Adversarial Networks». In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, June 2017, pp. 214–223. URL: https://proceedings.mlr.press/v70/arjovsky17a.html (cit. on p. 29).

[46] Christian Ledig et al. «Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network». In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 105–114. DOI: 10.1109/CVPR.2017.19 (cit. on p. 29).

[47] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. «Image-To-Image Translation With Conditional Adversarial Networks». In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2017 (cit. on pp. 29, 52, 53, 69, 70, 76).

[48] Xudong Mao, Qing Li, Haoran Xie, Raymond Y.K. Lau, Zhen Wang, and Stephen Paul Smolley. «Least Squares Generative Adversarial Networks». In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. Oct. 2017 (cit. on p. 29).

[49] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. «Progressive Growing of GANs for Improved Quality, Stability, and Variation». In: *CoRR* abs/1710.10196 (2017). arXiv: 1710.10196. URL: http://arxiv.org/abs/1710.10196 (cit. on p. 30).

[50] Mang Ye, Jianbing Shen, Gaojie Lin, Tao Xiang, Ling Shao, and Steven C. H. Hoi. «Deep Learning for Person Re-identification: A Survey and Outlook». In: *CoRR* abs/2001.04193 (2020). arXiv: 2001.04193. URL: https://arxiv.org/abs/2001.04193 (cit. on pp. 31–34).

[51] Liang Zheng, Yi Yang, and Alexander G. Hauptmann. «Person Re-identification: Past, Present and Future». In: *CoRR* abs/1610.02984 (2016). arXiv: 1610.02984. URL: http://arxiv.org/abs/1610.02984 (cit. on pp. 31, 39).

[52] Liang Zheng, Hengheng Zhang, Shaoyan Sun, Manmohan Chandraker, and Qi Tian. «Person Re-identification in the Wild». In: *arXiv preprint arXiv:1604.02531* (2016) (cit. on p. 32).

[53] Abir Das, Anirban Chakraborty, and Amit K Roy-Chowdhury. «Consistent Re-identification in a Camera Network». In: *European Conference on Computer Vision*. Vol. 8690. Lecture Notes in Computer Science. Zurich: Springer, 2014, pp. 330–345 (cit. on p. 33).

[54] Piotr Dollar, Christian Wojek, Bernt Schiele, and Pietro Perona. «Pedestrian detection: A benchmark». In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009, pp. 304–311. DOI: 10.1109/CVPR.2009.5206631 (cit. on p. 33).

[55] Pedro F. Felzenszwalb, Ross B. Girshick, David McAllester, and Deva Ramanan. «Object Detection with Discriminatively Trained Part-Based Models». In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32.9 (2010), pp. 1627–1645. DOI: 10.1109/TPAMI.2009.167 (cit. on pp. 33, 37).

[56] Zhun Zhong, Liang Zheng, Donglin Cao, and Shaozi Li. «Re-ranking Person Re-identification with k-reciprocal Encoding». In: (2017) (cit. on pp. 35, 77–80, 89, 90).

[57] Song Bai and Xiang Bai. «Sparse contextual activation for efficient visual re-ranking». In: *IEEE Transactions on Image Processing* (2016) (cit. on p. 35).

[58] Ergys Ristani, Francesco Solera, Roger Zou, Rita Cucchiara, and Carlo Tomasi. «Performance Measures and a Data Set for Multi-Target, Multi-Camera Tracking». In: *European Conference on Computer Vision workshop on Benchmarking Multi-Target Tracking*. 2016 (cit. on p. 37).

[59] Mengran Gou, Srikrishna Karanam, Wenqian Liu, Octavia Camps, and Richard J. Radke. «DukeMTMC4ReID: A Large-Scale Multi-camera Person Re-identification Dataset». In: *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2017, pp. 1425–1434. DOI: 10.1109/CVPRW.2017.185 (cit. on p. 37).

[60] M. Farenzena, L. Bazzani, A. Perina, V. Murino, and M. Cristani. «Person re-identification by symmetry-driven accumulation of local features». In: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 2010, pp. 2360–2367. DOI: 10.1109/CVPR.2010.5539926 (cit. on p. 39).

[61] Hao Luo, Youzhi Gu, Xingyu Liao, Shenqi Lai, and Wei Jiang. «Bag of Tricks and a Strong Baseline for Deep Person Re-Identification». In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2019, pp. 1487–1495. DOI: 10.1109/CVPRW.2019.00190 (cit. on pp. 40, 44, 66, 74, 75, 81–83).

[62] Yifan Sun, Liang Zheng, Yi Yang, Qi Tian, and Shengjin Wang. «Beyond Part Models: Person Retrieval with Refined Part Pooling (and A Strong Convolutional Baseline)». In: *Proceedings of the European Conference on Computer Vision (ECCV)*. Sept. 2018 (cit. on pp. 40, 41, 44, 45).

[63] Rodolfo Quispe and Helio Pedrini. «Top-DB-Net: Top DropBlock for Activation Enhancement in Person Re-Identification». In: *25th International Conference on Pattern Recognition* (2020) (cit. on pp. 41, 44, 45).

[64] Zijie Zhuang, Longhui Wei, Lingxi Xie, Tianyu Zhang, Hengheng Zhang, Haozhe Wu, Haizhou Ai, and Qi Tian. «Rethinking the Distribution Gap of Person Re-identification with Camera-Based Batch Normalization». In: *European Conference on Computer Vision*. Springer. 2020, pp. 140–157 (cit. on pp. 41, 44, 66, 67, 81, 82).

[65] Zhedong Zheng, Xiaodong Yang, Zhiding Yu, Liang Zheng, Yi Yang, and Jan Kautz. «Joint discriminative and generative learning for person re-identification». In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019 (cit. on pp. 42–44, 56).

[66] Yixiao Ge, Zhuowan Li, Haiyu Zhao, Guojun Yin, Shuai Yi, Xiaogang Wang, and hongsheng Li. «FD-GAN: Pose-guided Feature Distilling GAN for Robust Person Re-identification». In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Vol. 31. Curran Associates, Inc., 2018. URL: https://proceedings.neurips.cc/paper/2018/file/c5ab0bc60ac7929182aadd08703f1ec6-Paper.pdf (cit. on pp. 42, 44, 45).

[67] Guangcong Wang, Jianhuang Lai, Peigen Huang, and Xiaohua Xie. «Spatial-Temporal Person Re-identification». In: (2019), pp. 8933–8940 (cit. on p. 44).

[68] Joaquin Quionero-Candela, Masashi Sugiyama, Anton Schwaighofer, and Neil D. Lawrence. *Dataset Shift in Machine Learning*. The MIT Press, 2009. ISBN: 0262170051 (cit. on pp. 47, 48).

[69] Liangchen Song, Cheng Wang, Lefei Zhang, Bo Du, Qian Zhang, Chang Huang, and Xinggang Wang. «Unsupervised domain adaptive re-identification: Theory and practice». In: *Pattern Recognition* 102 (2020), p. 107173. ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2019.107173. URL: https://www.sciencedirect.com/science/article/pii/S003132031930473X (cit. on p. 49).

[70] Garrett Wilson and Diane J. Cook. «A Survey of Unsupervised Deep Domain Adaptation». In: *ACM Trans. Intell. Syst. Technol.* 11.5 (July 2020). ISSN: 2157-6904. DOI: 10.1145/3400066. URL: https://doi.org/10.1145/3400066 (cit. on pp. 50, 51).

[71] Mei Wang and Weihong Deng. «Deep visual domain adaptation: A survey». In: *Neurocomputing* 312 (2018), pp. 135–153. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2018.05.083. URL: https://www.sciencedirect.com/science/article/pii/S0925231218306684 (cit. on pp. 50, 51).

[72] Yanghao Li, Naiyan Wang, Jiaying Liu, and Xiaodi Hou. «Demystifying Neural Style Transfer». In: *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*. 2017, pp. 2230–2236. DOI: 10.24963/ijcai.2017/310. URL: https://doi.org/10.24963/ijcai.2017/310 (cit. on pp. 50, 52).

[73] Muhammad Ghifary, W. Bastiaan Kleijn, Mengjie Zhang, David Balduzzi, and Wen Li. «Deep Reconstruction-Classification Networks for Unsupervised Domain Adaptation». In: *Computer Vision – ECCV 2016*. Ed. by Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling. Cham: Springer International Publishing, 2016, pp. 597–613 (cit. on p. 50).

[74] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario March, and Victor Lempitsky. «Domain-Adversarial Training of Neural Networks». In: *Journal of Machine Learning Research* 17.59 (2016), pp. 1–35. URL: http://jmlr.org/papers/v17/15-239.html (cit. on p. 50).

[75] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. «Unsupervised Representation Learning by Predicting Image Rotations». In: *International Conference on Learning Representations*. 2018. URL: https://openreview.net/forum?id=S1v4N2l0- (cit. on p. 51).

[76] Fabio Maria Carlucci, Antonio D'Innocente, Silvia Bucci, Barbara Caputo, and Tatiana Tommasi. «Domain Generalization by Solving Jigsaw Puzzles». In: *CVPR*. 2019 (cit. on p. 51).

[77] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. «Image Style Transfer Using Convolutional Neural Networks». In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 2414–2423. DOI: 10.1109/CVPR.2016.265 (cit. on pp. 52, 59, 60, 70).

[78] Haotian Tang, Yiru Zhao, and Hongtao Lu. «Unsupervised Person Re-Identification With Iterative Self-Supervised Domain Adaptation». In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2019, pp. 1536–1543. DOI: 10.1109/CVPRW.2019.00195 (cit. on pp. 54, 56, 59, 60, 73, 81, 82).

[79] Yang Zou, Xiaodong Yang, Zhiding Yu, Bhagavatula Vijayakumar, and Jan Kautz. «Joint disentangling and adaptation for cross-domain person re-identification». In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2020 (cit. on pp. 54, 56, 57, 59, 60, 73, 81, 82, 86).

[80] Hao Chen, Yaohui Wang, Benoit Lagadec, Antitza Dantcheva, and Francois Bremond. «Joint Generative and Contrastive Learning for Unsupervised Person Re-Identification». In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2021, pp. 2004–2013 (cit. on pp. 54, 57, 59, 60, 73, 81, 82, 86).

[81] Yang Fu, Yunchao Wei, Guanshuo Wang, Yuqian Zhou, Honghui Shi, and Thomas S. Huang. «Self-Similarity Grouping: A Simple Unsupervised Cross Domain Adaptation Approach for Person Re-Identification». In: *The IEEE International Conference on Computer Vision (ICCV)*. Oct. 2019 (cit. on pp. 54, 59, 60).

[82] Yixiao Ge, Dapeng Chen, and Hongsheng Li. «Mutual Mean-Teaching: Pseudo Label Refinery for Unsupervised Domain Adaptation on Person Re-identification». In: *International Conference on Learning Representations*. 2020. URL: https://openreview.net/forum?id=rJlnOhVYPS (cit. on pp. 55, 59, 60).

[83] Wenhao Wang, Fang Zhao, Shengcai Liao, and Ling Shao. «Attentive WaveBlock: Complementarity-enhanced Mutual Networks for Unsupervised Domain Adaptation in Person Re-identification and Beyond». In: *IEEE Transactions on Image Processing* (2022) (cit. on pp. 55, 59, 60).

[84] Xiaobin Liu and Shiliang Zhang. «Domain Adaptive Person Re-Identification via Coupling Optimization». In: *Proceedings of the 28th ACM International Conference on Multimedia*. New York, NY, USA: Association for Computing Machinery, 2020, pp. 547–555. ISBN: 9781450379885. URL: https://doi.org/10.1145/3394171.3413904 (cit. on pp. 55, 59, 60).

[85] Longhui Wei, Shiliang Zhang, Wen Gao, and Qi Tian. «Person Transfer GAN to Bridge Domain Gap for Person Re-identification». In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2018), pp. 79–88 (cit. on p. 56).

[86] Weijian Deng, Liang Zheng, Qixiang Ye, Guoliang Kang, Yi Yang, and Jianbin Jiao. «Image-Image Domain Adaptation with Preserved Self-Similarity and Domain-Dissimilarity for Person Re-identification». In: *CVPR*. 2018 (cit. on pp. 56, 59, 60, 73, 81, 82).

[87] Yanbei Chen, Xiatian Zhu, and Shaogang Gong. «Instance-Guided Context Rendering for Cross-Domain Person Re-Identification». In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019, pp. 232–242. DOI: 10.1109/ICCV.2019.00032 (cit. on pp. 57–60, 81, 82).

[88] Jianing Li and Shiliang Zhang. «Joint Visual and Temporal Consistency for Unsupervised Domain Adaptive Person Re-Identification». In: *ECCV*. 2020 (cit. on pp. 58–60, 66, 67, 81, 82).

[89] Tianyu Zhang, Lingxi Xie, Longhui Wei, Zijie Zhuang, Yongfei Zhang, Bo Li, and Qi Tian. «UnrealPerson: An Adaptive Pipeline towards Costless Person Re-identification». In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021 (cit. on pp. 60, 63, 64, 66, 67, 81–83).

[90] Matteo Fabbri, Guillem Brasó, Gianluca Maugeri, Aljoša Ošep, Riccardo Gasparini, Orcun Cetintas, Simone Calderara, Laura Leal-Taixé, and Rita Cucchiara. «MOTSynth: How Can Synthetic Data Help Pedestrian Detection and Tracking?» In: *International Conference on Computer Vision (ICCV)*. 2021 (cit. on pp. 60, 63).

[91] Sławomir Bąk, Peter Carr, and Jean-François Lalonde. «Domain Adaptation through Synthesis for Unsupervised Person Re-identification». In: *ECCV*. 2018 (cit. on pp. 61, 62, 64–66, 81).

[92] Epic Games Incorporated. *Unreal Engine*. 2021. URL: `https://www.unrealengine.com` (cit. on pp. 61, 63, 66).

[93] Xiaoxiao Sun and Liang Zheng. «Dissecting Person Re-identification from the Viewpoint of Viewpoint». In: *CVPR*. 2019 (cit. on pp. 61, 62, 64).

[94] Unity Technologies. *Unity Real-Time Development Platform*. 2021. URL: `https://unity.com` (cit. on pp. 61, 62).

[95] Yanan Wang, Shengcai Liao, and Ling Shao. «Surpassing Real-World Source Training Data: Random 3D Characters for Generalizable Person Re-Identification». In: *28th ACM International Conference on Multimedia (ACMMM)*. 2020 (cit. on pp. 61, 62, 64, 66, 67, 81–83, 86).

[96] MakeHuman Community. *MakeHuman: Open Source Tool for Making 3D Characters*. 2021. URL: `http://www.makehumancommunity.org` (cit. on p. 61).

[97] Igor Barros Barbosa, Marco Cristani, Barbara Caputo, Aleksander Rognhaugen, and Theoharis Theoharis. «Looking beyond appearances: Synthetic training data for deep CNNs in re-identification». In: *Computer Vision and Image Understanding* 167 (2018), pp. 50–62. ISSN: 1077-3142. DOI: `https://doi.org/10.1016/j.cviu.2017.12.002`. URL: `https://www.sciencedirect.com/science/article/pii/S1077314217302254` (cit. on pp. 62, 65).

[98] Rockstar North, Rockstar Leeds, and Rockstar Games. *Grand Theft Auto V*. 2021. URL: `https://www.rockstargames.com/gta-v` (cit. on p. 63).

[99] Matteo Fabbri, Fabio Lanzi, Simone Calderara, Andrea Palazzi, Roberto Vezzani, and Rita Cucchiara. «Learning to Detect and Track Visible and Occluded Body Joints in a Virtual World». In: *European Conference on Computer Vision (ECCV)*. 2018 (cit. on pp. 63, 64).

[100] Stephan R. Richter, Hassan Abu AlHaija, and Vladlen Koltun. «Enhancing Photorealism Enhancement». In: *arXiv:2105.04619* (2021) (cit. on p. 63).

[101] Aäron van den Oord, Yazhe Li, and Oriol Vinyals. «Representation Learning with Contrastive Predictive Coding». In: *CoRR* abs/1807.03748 (2018). arXiv: 1807.03748. URL: http://arxiv.org/abs/1807.03748 (cit. on pp. 71, 72).

[102] Tim Salimans, Ian J. Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. «Improved Techniques for Training GANs». In: *CoRR* abs/1606.03498 (2016). arXiv: 1606.03498. URL: http://arxiv.org/abs/1606.03498 (cit. on p. 91).