PH.D. SCHOOL
UNIVERSITY OF MILANO-BICOCCA

Department of Informatics, Systems and Communication
PhD Program in Computer Science - XXXI Cycle

Visual Anomaly Detection for Automatic Quality Control

Ph.D. Dissertation of: Flavio Piccoli

Supervisor: Prof. Raimondo Schettini
Co-Supervisor: Dr. Paolo Napoletano
Tutor: Prof. Alberto Leporati
Ph.D. Coordinator: Prof.ssa Stefania Bandini

Academic Year 2018-2019

Acknowledgements

This thesis is dedicated to my parents Luciano and Marisa, that always supported me. To my granparents Adriano, Felicita and Concetta for transmitting me the love for life. To all my friends, who made me smile in the bad moments. To Irene, my love.

To my supervisor Raimondo Schettini for believing in me also when I didn't deserve it. To Paolo Napoletano for his patience and his precious suggestions; to Simone Bianco and Claudio Cusano for their support.

To my collegues Luigi Celona, Marco Buzzelli and Davide Mazzini for the fraternal team spirit that makes me feel at home when I am in lab. To the guys from IVL and other labs with which I shared many joyful moments during lunch time.

To my acquired collegues Fabio, Vincenzo, Marco, Alessandro, Daniele and Matteo for allowing me to do this PhD, for getting me in contact with the industrial environment and for the good moments spent around the world.

Abstract

Automatic quality control is one of the key ingredients for the fourth industrial revolution that will lead to the development of the so called industry 4.0. In this context, a crucial element is a production-compatible-time detection of defects, anomalies or product failures. This thesis focuses exactly on this theme: anomaly detection for industrial quality inspection, ensured through the analysis of images depicting the product under inspection. This analysis will be done through the use of machine learning, and especially through the use of convolutional neural networks (CNNs), a powerful instrument used in image analysis. This thesis starts with an extensive study on the subject to introduce the reader and to propose a pipeline for automatic anomaly detection. This pipeline is composed by two steps: 1) the enhancement of the input images for highlighting defects; 2) the detection of the anomalies.

The first step is addressed with the use of a global color transformation able to remove undesired light effects and to enhance the contrast. This transformation is inferred through the use of SpliNet, a new CNN-based method here presented, that is able to enhance the input images by inferring the parameters of a set of splines.

In the context of anomaly detection, two methods are presented. The first one has the aim of modeling normality by learning a dictionary and using it in test time to determine the degree of abnormality of an inquiry image. This method is based on deep learning, which is known to be data-hungry. However, the proposed algorithm is able to work also on very small trainsets (in the order of five images). The presented method boosts the performances of 5% with respect to the state-of-the art for the SEM-acquired nanofibers dataset, achieving an area under curve of 97.4%. The second proposed algorithm is a generative method able to restore the input, creating an anomaly-free version of the inquiry image. This method uses a set of local transforms to restore the input images. Specifically, these transforms are sets of polynomials of degree two, whose parameters are determined through the use of a convolutional neural network. In this context, the method can be tuned with a parameter toward accuracy or speed, for matching the needs of the final user.

To address the lack of data that is suffered in this field, a totally new method for data augmentation based on deep learning is presented. This method is able to generate thousands of new synthesized samples starting from a few and thus is particularly suitable for augmenting long-tail datasets. The quality of the synthesized samples is demonstrated by showing the increase in performance of machine learning algorithms trained on the augmented dataset. This method has been employed to enlarge a dataset of defective asphalts. In this context, the use of the augmented dataset permitted to increase the average performance on anomaly segmentation of up to 17.5 percentage points. In the case of classes having a low cardinality, the improvement is up to 54.5 percentage points. For all the methods here presented I show their effectiveness by analyzing the results with the respective state-of-the-art and show their ability in outperforming the existing methods.

Table of contents

List of figures

List of tables

Part I

Background

Chapter 1

Introduction

"We will be restoring normality just as soon as we are sure what is normal."

Douglas Adams,
The Hitchhiker's Guide to the Galaxy

Industrial quality control is a broad discipline that touches many domains such as phisics, chemistry, economy, psicology and computer vision. According to the International Organization for Standardization (ISO), *quality control* is a process by which entities review the quality of all factors involved in production. ISO 9000 [58] defines quality control as "A part of quality management focused on fulfilling quality requirements". This discipline regards many aspects of the industrial environment, from the design of the production chains to the procedures adopted by all the actors directly and indirectly involved in the production. In particular, the most important aspect of the quality control is the direct analysis of the products along the production chain.

Cyber-Physical systems [66], Internet of Things [119], Cloud Computing [21] and Cognitive Computing [11] are the key ingredients of the fourth industrial revolution that will lead to development of the so called Industry 4.0 [44, 72]. The industry of future is a "smart factory" that integrates new hardware, software and communication technologies to obtain smart production processes that increase productivity and reduce costs in manufacturing environments. One of the challenges of the Industry 4.0 designers is the optimization of the manufacturing processes. A key element to this aim is the early, or production-compatible-time, detection of defects or anomalies and production failures. This enables the prevention of production errors thus increasing productivity, quality, and consequently leading to a economic benefit for the factory [44].

Defect or anomaly detection in industry is performed in several ways: 1) manually by humans that monitor the entire production process; 2) automatically by a computer-

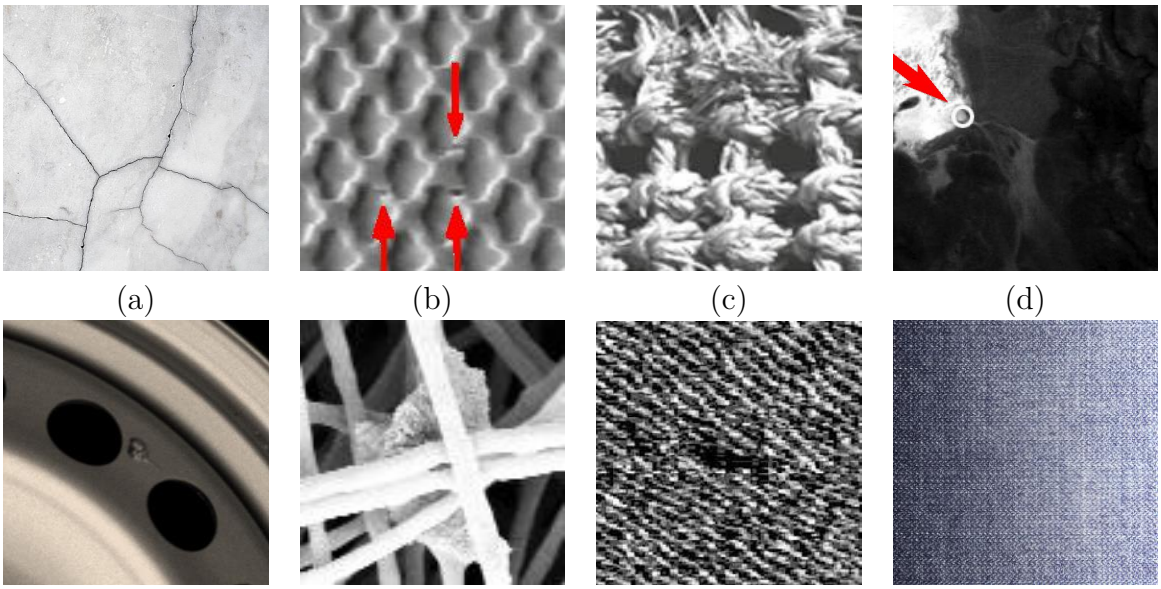


Fig. 1.1 Examples of anomalies. (a) shows cracs in a marble, (b) holes in an electronic wafer, (c) a broken fabric, (d) an anomaly in a spectrometer acquisition, (e) a hole in a wheel, (f) excess of material in a nanofiber, (g) a mine on the bottom of the sea (h) an unexpected fade on jeans

based system that monitors, mostly with the help of digital cameras or sensors, the production process; 3) semi-automatically by human that interacts with a computer-based defect detection system [69, 70, 27, 117]. In all ways, defect detection is performed in two different moments: during or at the end of the production process. To take the chance to correct the production process, defect detection should be performed during the production process in a time that should be less, or at most comparable with, the production time itself. This time constraint permits to provide feedbacks or alarms that may be used to correct the production process [69].

With the term *anomaly* one refers to anything that deviates from normality (see section 1.1 for further details), therefore one does not define an explicit model for the anomalies. Differently, with the term *defect* one indicates a member of a sub-group of anomalies that share common properties and therefore they can be described with an explicit model. Figure 1.1 shows some examples of anomalies. A real challenge is exactly to design and develop methods that will be able to identify deviations from normality that do not have an explicit model. Methods based on this approach will be more flexible and more adapt to be used in real contexts, where the alterations that can occur to the products during the production line are not kown *a-priori* (or partially known). An example of this concept can be a production line of fabrics. As it's possible to see

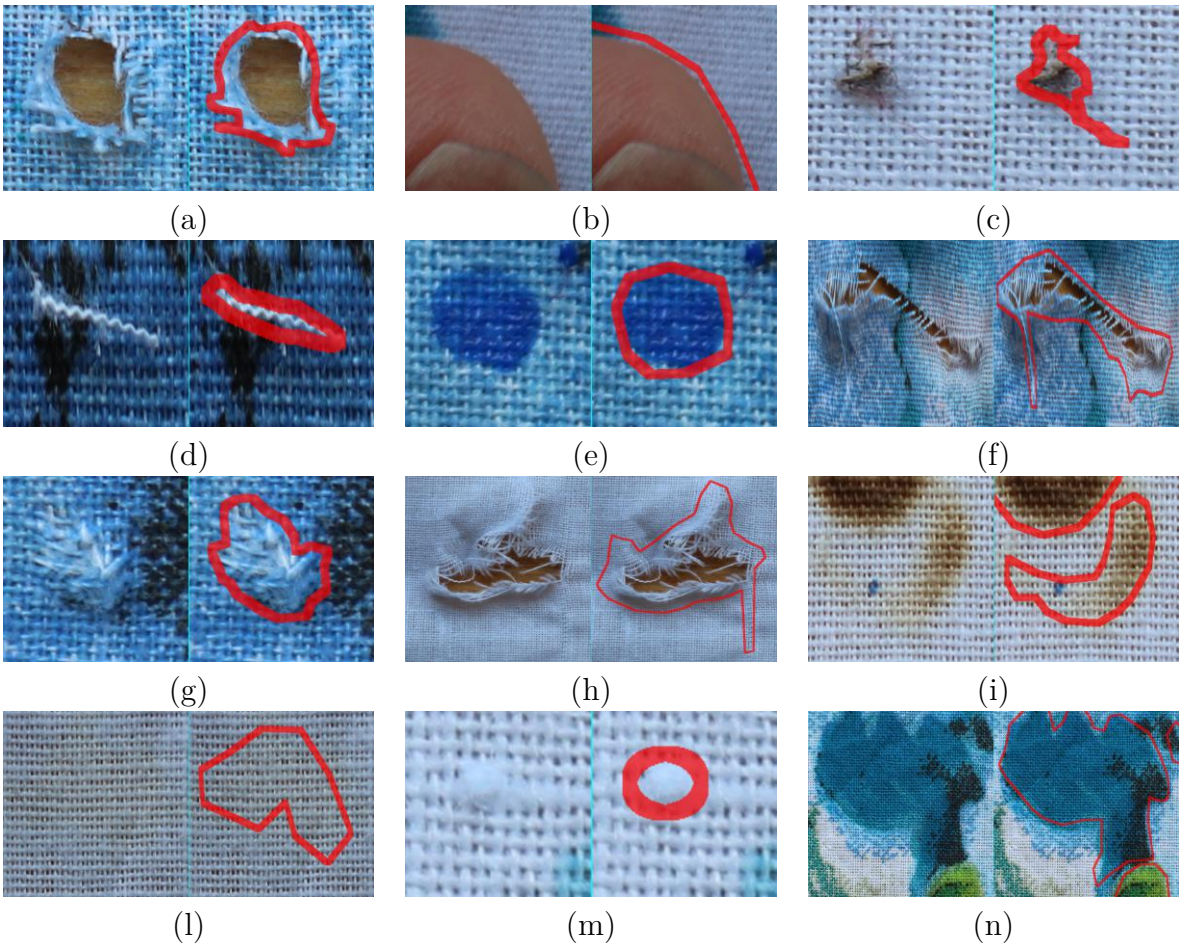


Fig. 1.2 Examples of anomalies and defects on fabric. (a) hole, (b)(c) external corps, (d) extra material, (e)(n) stain, (f)(h) tear, (g) broken fabric, (i) burn, (l) tissue discoloration, (m) floating woven wire

in figure 1.2, many anomalies and defects can occur during the production process. If in design-time only holes, stains and tears were expected as a defective class, a defect detection method will fail to detect extra material or any type of external corp, while an anomaly detection system will be able to detect everything. This is because a defect detection system will model the three defective classes, while the anomaly detection system will model the fabric.

Methods for anomaly detection will be investigated in this thesis. In particular, the focus will be on detection of anomalies through the analysis of images depicting the product under inspection. This analysis will be done through the use of machine learning techniques, in particular through the use of *Convolutional Neural Networks (CNNs)* [73], which are a powerful instrument widely used in image analysis. In this context, anomaly detection is a way more challenging then defect detection, because it resides in

the groups of algorithms of *one-class learning*. In fact, object detection algorithms learn the the appearance and the structures of the objects' classes by analyzing the intra-class differences, condition that does not hold in *one-class learning*.

1.1 What is an anomaly?

According to the Oxford dictionary [1], an anomaly is defined as:

> "Something that deviates from what is standard, normal, or expected."

In the case of industrial quality inspection (IQI), an anomaly is a phisical, chemical, mechanical variation from normality that is present on the product under inspection. The detection of an anomaly sometimes can be really hard or - in some extreme cases - impossible to detect. In the case of industrial quality inspection ensured through the analysis of images, an anomaly is defined as a visual discrepancy of a portion or of the entire product in the corresponding image. Depending from the type of product and the desired quality level, slight variations in some case are accepted. For example, in the fabric production, a cloth can be a little bit faded with respect to the forseen colors. According to the definition, there is no assumption on the structure, shape and on the visual apperance of an anomaly, therefore is assumed that an anomaly can be anything.

There are two different types of anomalies. The first ones are called *highly localized anomalies* while the second one are the *spatially distributed anomalies*. In the first group fall all the anomalies that affect a portion of the product under inspection that are highly localized. For instance, it's possible to determine if this type of anomaly is present in the product by just looking a sub-portion of the corresponding image. In the *spatially distributed anomalies* fall all those anomalies that can be found only by looking at the entire picture at the same time. To better understand the difference between these two classes of anomalies, see figure 1.1. The anomalies present in the examples from (a) to (g) can be easily detected and marked with a red pen, while the unexpected fade in example (h) cannot be separated from the jeans fabric on which it appears. This hintrinsic difference among the two types of anomalies is reflected in the way anomalies are represented in machine learning. *Highly localized* anomalies are indicated through the use of a binary class where the white pixels indicate the anomalous area (while the black the normal one). Differently, samples containing *spatially distributed anomalies* are paired to anomaly-free images of the same product. Some researchers can wrongly think that bounding boxes can be used to describe *Highly localized* anomalies. However, since in anomaly detection we do not explicitly model the anomalies and thus we do not make

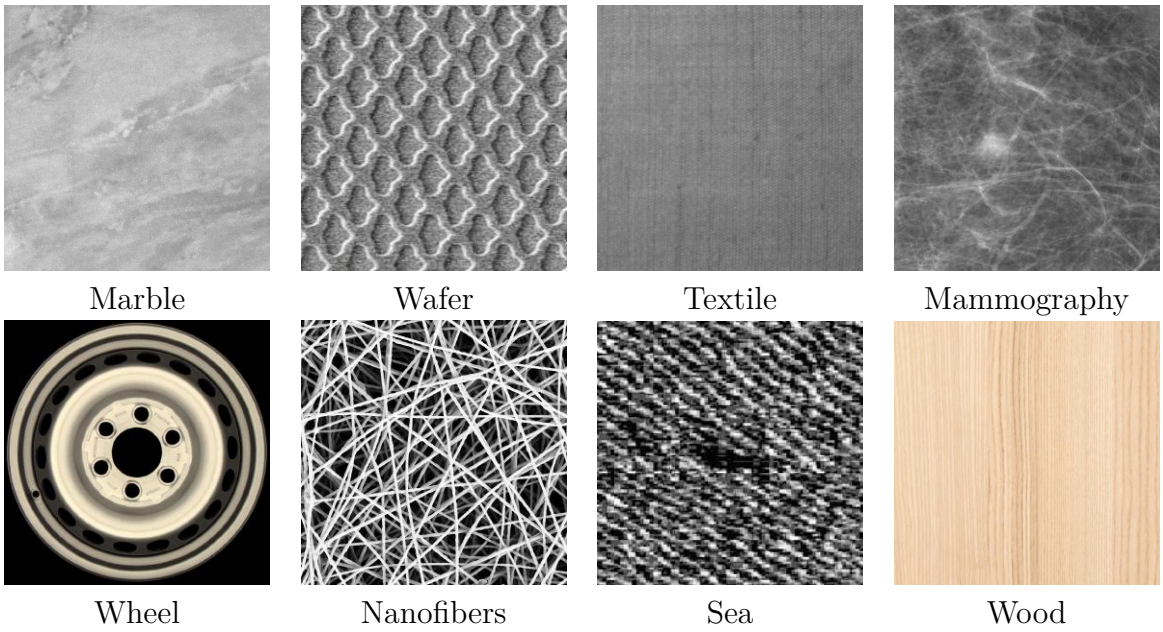


Fig. 1.3 Examples of normal signals (anomaly-free samples).

any assumption on their apperance (like it would occur in a task of object detection), this will result in a wrong choice.

1.2 The normal signal (background)

With the term *normal signal* [38], we indicate the images corresponding to the products in analysis that do not present anomalies. These images represent the normality and depend of course from the type and the nature of the product under investigation. In a context of anomaly detection, the *normal signal* is also called *background*, because it's the part of the signal that we want to detect and discard during the analysis. What remains then, is the *forground*, i.e. the anomalies that we want to detect. The anomaly-free images that compose the *normality*, can be *homogeneous* or *heterogeneous*. In the case of *homogeneous* samples, the pattern of the background is always the same, except for slight allowed variations. In this case, we have repetitive patterns that compose the aspect of the product. If these patterns have a specific frequency (or reside in a frequency range) and can be easily identified in the frequency domain, they are called *frequency separable*. In contrast, in the case of *heterogeneous* samples, the background can be composed by structures that are spatially varying and there isn't repetitivity among the different portions of the signal. This last type of normal signal is harder to detect because it's not repetitive and therefore each sample is different from the others. Figure 1.3 shows some

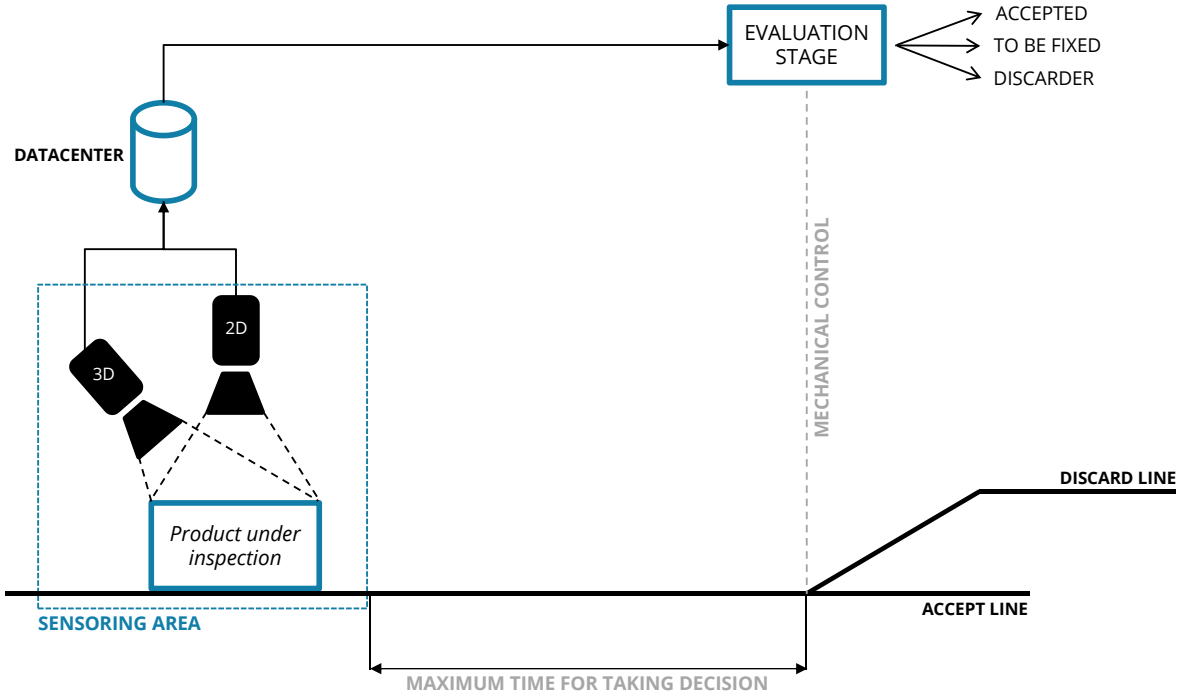


Fig. 1.4 Pipeline for anomaly detection in an industrial production chain.

examples of anomaly-free samples. The nature of the background affects the choice of the anomaly detection algorithm (for further details, see chapter 5).

1.3 Possible pipeline for anomaly detection

A possible pipeline to detect anomalies is the one depicted in figure 1.4. The product under inspection that is coming out from an intermediate or the final processing stage goes into a sensing area through a conveyor belt (black line on the bottom). In this place a set of mechanical, chemical and/or phisical threatments are applied to the product with the aim of highlighting possible defects and anomalies that otherwise wouldn't be visible. The sensing area is equipped with a set of acquisition devices that produce a set of images that will be later analyzed by the evaluation stage. The types of devices depend from the dimension of the product and the level of analysis. For instance, if the material under inspection is of nanometric size, it can be adopted a Scanning Electron Microscope (SEM), while if it's required to control the quality of fabrics, it's possible to adopt a high-definition camera. The acquisition technology (2D, 2.5D, 3D, visible light, mono, stereo, laser, x-ray, eco, thermocamera) depends from the characteristics that we want to observe. For example, the use of a thermocamera will highlight thermal properties of the

product under inspection and so on. Sometimes is necessary to adopt multiple devices of the same type to cover the entire product area and then match the corresponding images. Once the product is acquired, the produced images are transmitted to a datacenter, usually redundant, and are later read by the evaluation stage. This last stage accesses to the images stored in the datacenter and processes them with the aim to define weather the product under inspection has to be accepted, must be discarded or it just needs to be fixed. The evaluation stage is connected to a mechanical switch acting on the conveyor belt. In this way, if a product does not pass the quality control, it will be dispached in the discard line. A decision on the product must take place in a short time. Specifically, this time depends from the distance d between the sensing area and the mechanical dispatcher controlled by the evaluation stage (see figure 1.4). Let s be the speed to which the product is moving inside the production chain. Then, the maximum time for taking a decision is:

$$t_{max} = \frac{s}{d} \tag{1.1}$$

This thesis proposes a pipeline and different algorithms that can be adopted in the evaluation stage in order to address the task of anomaly detection.

1.4 Sources of anomalies

An anomaly detection system does not take care only of the anomalies that can occur to the products during the product chain. It has to take into account all possible anomalies that can happen during the acquisition and transmission process. This is because in most of the cases, there are no mechanisms and feedbacks to track down and identify the source of the anomaly. Therefore, the evaluation stage must signal the anomaly in any way, since there are no guarantees that the product is normal. There are multiple issues, related or not to the quality of the product, that can occur during the decision chain. Specifically:

- *Product Issues.* They can be a product dissimilarity from the orginal design or the can come from an external interference, such as an external corp that gets stuck or lie on the product.
- *Acquisition Issues.* Even though the acquisition is usually done in a closed environment, it can happen that a unwanted light spot shows up ruining the acquisition conditions. This can also happen if some liquid contaminates the product, making it more shiny. In this step, also a light failure or an unmodelled noise sensor can produce anomalies.

- *Transmission Issues.* Transmission cables are subject to electromagnetic interference (EMI), also called radio-frequency interference (RFI) and/or power surges that can alter the data transmitted. High usages of the transmission pipelines can overflow the throughput of the cables and mess up the passing data. These issues are usually the ones that can be detected by using error detection techniques such as parity bits, repetition codes, checksums, cyclic redundancy checks, cryphographic has functions and error-correcting codes.

- *Storage Issues.* Storage failures or lossy data compression can also alter the produced images.

- *Evaluation Issues.* The model complexity can be such that the system is overfitted or underfitted. Furthermore, the model can work better for some products and worse for others, allowing some anomalies to go through without being detected. In safe-critical productions, the models are recall-oriented to avoid the placing on market of defective products that can cause death or injuries of the clients. A recall-oriented method however, can produce a lot of false positives that can make the second operator under-estimate the reporting of the anomaly detection system and thus making it unuseful.

1.5 Human-issues in detecting anomalies

Anomaly detection operated by a human operator can be affected by one or more factors that can change the final outcome of the evaluation [113]. This can determine the miss detection of an important anomaly or the detection of false positives, depending from the status of the operator. These factors can be time- or space-varing. The time-varing factors can loop in a time range or not, and they can interest a long or a short time range. These issues can be cathegorized depending from their nature. In particular, there are:

- *Intra-personal factors.* Experts that evaluate the quality of the products are usually subject to changes in their detection accuracy and evaluation criteria; for example at the end of the day are more tired and have a decrease in the performances or during time their experience increases making their ability more robust and accurate.

- *Inter-personal factors.* Humans encharged to perform quality control usually exchange their experience. This fact contributes to make the quality level quite

similar among experts, however it makes the outcome of the analysis different day by day, and in some cases introduces wrong knowledge to the evaluators.

- *Boundary-uncertainty factor.* A quality operator is usually very good in doing its job and spot anomalies in the product; however she/he becomes less accurate when the task slightly changes and she/he is required to segment the found anomaly. This is not because the operator is not good but because finding boundaries between the anomalous part and the normal region is intrinsically hard. Imagine a scratch that is very soft at the beginning and little by little goes deeper until it becomes a cut. Suppose that in this case the scratches are allowed but the cuts no. It becomes really hard to define a boundary after which this scratch becomes an anomaly without further information except the visual inspection.

- *Multiple-detection factor.* When the person encharged to control detects an anomaly, for him the product it's discarded. Therefore she/he does not proceed in the analysis of the product. This can be a problem when the found anomaly is fixable but there are other anomalies that weren't found. Another problem is the relativity between the found anomalies. The degree of severity associated by the expert to the found anomalies changes when the anomalies are found together because unavoidably the human perception makes comparisons between objects in the field of view.

- *Point-of-view factors.* The analysis of a product is affected by the position of the observer with respect to the product under analysis. In particular, the distance of the observer changes its perception of the gravity of a possible anomaly. If the case of a direct analysis on the product, this effect is usually discounted by changing the viewing position. At the countrary, if the analysis is indirect and it's done through the analysis of images, the distance in this case corresponds to the zoom. This is a problem because evaluation at different scales gives different outcomes.

- *Product manipulation.* Phisical manipulations are in some cases very useful to highlight possible defects, however some products require a big effort for the quality inspector and when she/he get tired, these manipulations become less useful. This is a problem because it means that by the end of the day the number of miss-detections increases.

1.6 Implications of adopting an automatic anomaly detection system

A company that inserts in its production chain an automatic detection system has some implications that affect different aspects of the industrial ecosystem [88][64][72][106].

- *Better job quality.* The life of a quality inspector it's really hard, because usually the spots where she/he has to stay in order to accomplish this task are inside the production chain and therefore they are small and confined places in cramped conditions. The sobstitution with an automatic system would cause a change of their work job inside the company, increasing the resources in another task in the production and therefore the increase of their job and life quality.

- *Lower prices.* Quality control made by experts is expansive in terms of time (and therefore money). The analysis of the quality in some contexts is considered a bottleneck because the production chain in that part has to slow down the process. An automatic system can save a lot of time and money removing this bottleneck and therefore lower the price of the final product. Some companies do not check the quality of all the products but only a subset to avoid this waste of time. Here again, there is a loss of money because if a stock of products goes out in the market with some problems, there are many costs that the company has to afford. The more a product can affect the security of the persons and the more will be the cost of an anomaly on a product that goes in the market. Another problem of doing quality control only on a subset is that if a problem affects the production chain (for example a machinery has a problem and causes anomalies on the products), the signaling of this event is not immediate and therefore a big waste of raw material occurs, increasing the final price of the single product.

- *Higher product quality.* An automatic system never gets tired and performs better than the human quality control. Therefore the quality of the produced products increases.

- *Definition of a standard.* Many factors of a human-based quality control such as experience, the worker status, the culture, etc. make the decisions uneven. The same product, presented to different experts (or to the same expert in different times), will be judged differently. The adoption of an automatic system gets rid of all those ambiguities and *de facto* creates a quality standard.

- *Easyness of broadcasting new quality standards.* An update of the quality standard in an automatic system means a transmission of the software encharged to evaluate the products. A software release it's quick and inexpansive. Without the automatic ADS, a new way to evaluate the products must be broadcasted between quality control workers, therefore it's required to make meetings and tutorials for the updates. If it's a worldwide company, this means removing experts from the production chain and send them to around the world to update other teams.

1.7 Lack of data

The acquisition technology (2D, 2.5D, 3D, visible light, mono, stereo, laser, x-ray, eco, thermocamera), the acquisition setup (mechanical, electrical or chemical treatments to highlight defects) and the acquisition conditions (position of the camera, type of light, acquisition environment) that characterize the sensing area are some of the most important characteristic of an industrial quality inspection system and depend from the type of manufact produced, the location and the structure of the production plant and finally from the quality level that it's required. Those aspects, in combination with the anomaly detection methods, are kept secret from the companies because their study and developement is very expensive and gives a lot of advantage on the market. For this reason, for the task of industrial quality inspection in computer vision, there is a huge lack of publicy available datasets. This is because if a competitor takes possession of a company's dataset, is able to determine the stages monitored along the production process, the acquisition technology, setup and conditions. Furthermore they can compare their quality with the company's obtained quality. This operation is called *reverse engineering* [40][39].

In this thesis, it will be proposed a method for data augmentation that will help researchers and engineers to create robust and accurate anomaly detection systems starting from a very small set of images. It will be shown that standard methods for data augmentation are not sufficient in the case of *long-tail* datasets. The majority of datasets for anomaly detection have a long-tail, because defects and anomalies occur more then others. For example, a machinery with a known issue will create a way more anomalies than external corps falling in the production line.

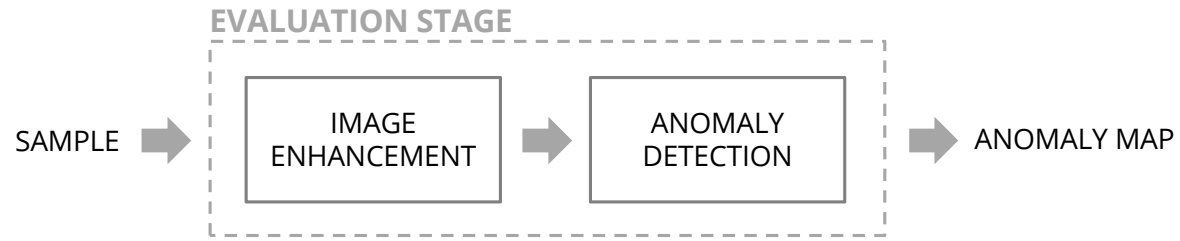


Fig. 1.5 The evaluation stage is composed by two main steps: the enhancement of the input images, which has the aim of highlighting anomalies and the anomaly detection step, which is encharged to find *highly localized* and *spatially distributed* anomalies.

1.8 Structure of this thesis

In this thesis we will talk about the evaluation stage (see section 1.3 for further details). A possible composition of the evaluation stage is the one depicted in figure 1.5. The steps of which is composed the evaluation stage are two: the preprocessing of the input images, here called *image enhancement* and the detection of the spatially distributed and highly localized anomalies. Each one of these steps has a different ground-truth type (see section 1.1) and different evaluation metrics. For this reason, these steps are treated and benchmarked separately. The thesis is organized as follows: firstly it will be introduced in section 2 the adversarial training, used later for the generation of synthetic data. This part composes the background needed to understand the following chapters. Afterwards, the problem of the lack of public data that affects the field of anomaly detection is addressed in part II by introducing a novel method for data augmentation. Then, there are two parts: part III that addresses the issue of enhancing the images and part IV, containing the proposed methods for detecting anomalies. This last part is divided in three chapters. The first one will give an introduction to the state of the art regarding anomaly detection. The second will talk about feature-based methods for anomaly detection (section 6), that are useful to detect the highly-localized anomalies; while the last one (section 7) will talk about generative methods, which are used to detect spatially-distributed anomalies.

Chapter 2

Adversarial Training

"You must not fight too often with one enemy, or you will teach him all your art of war."

Napoleon Bonaparte

Adversarial training is a special training configuration used to train a generative model. In this setup, during training time, a further neural network called *discriminator* or *critic*, is used to evaluate and improve the goodness of the generated results. The use of a neural network as a training loss for another network generates better results with respect to pointwise or local-operator-based losses such $L1$, $L2$ or $SSIM$ [55]. Sometimes this loss is also called *perceptual loss* [75] because it keeps into account higher level features that are learned during training that promote better results. This loss, in contrast with pointwise losses, generates more realistic and sharp contents. Pointwise losses in fact, promote blurriness [124]. The two networks, the *generator* and the *discriminator* compete against each other. Namely, at each step, the generator tries to fool the discriminator by creating always more realistic content, while the discriminator has to find useful features to distinguish between a real sample and a fake sample. The features found by the *discriminator* are used by the generator to promote the quality of the synthetic results. Figure 2.1 shows this particular training setup. Generators trained with this particular setup are called *Generative Adversarial Networks (GANs)*. Formally, GANs are semi-supervised algorithms [93], because they don't require paired samples. In fact, the task of the generator is to project the input into the manifold of the desired outputs and this is obtained by matching the features in the latent space given by the *discriminator*. Depending from the loss adopted in this context and from the structure of the *discriminator*, there are several variations of the GANs, in the next sub-chapters we will present the most important GANs that represent the state of the art. This chapter

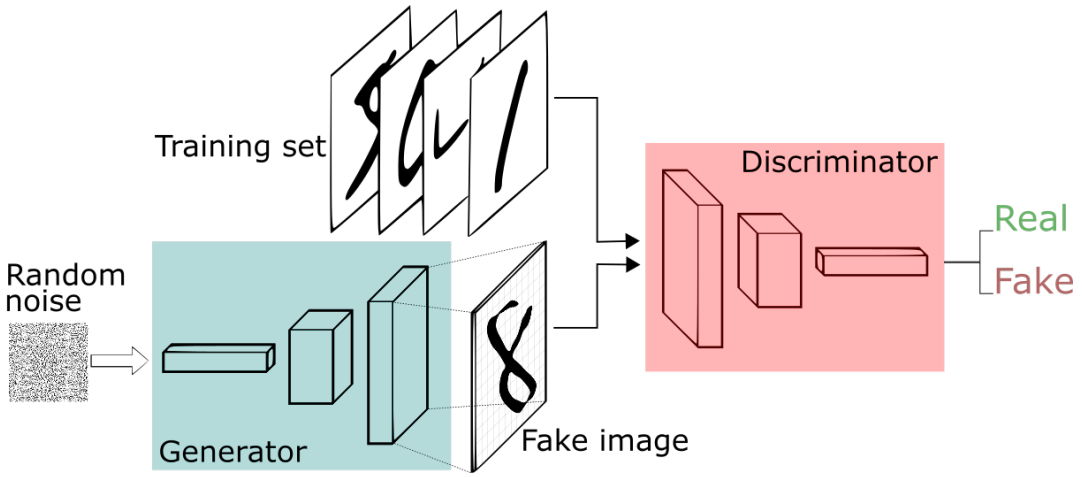
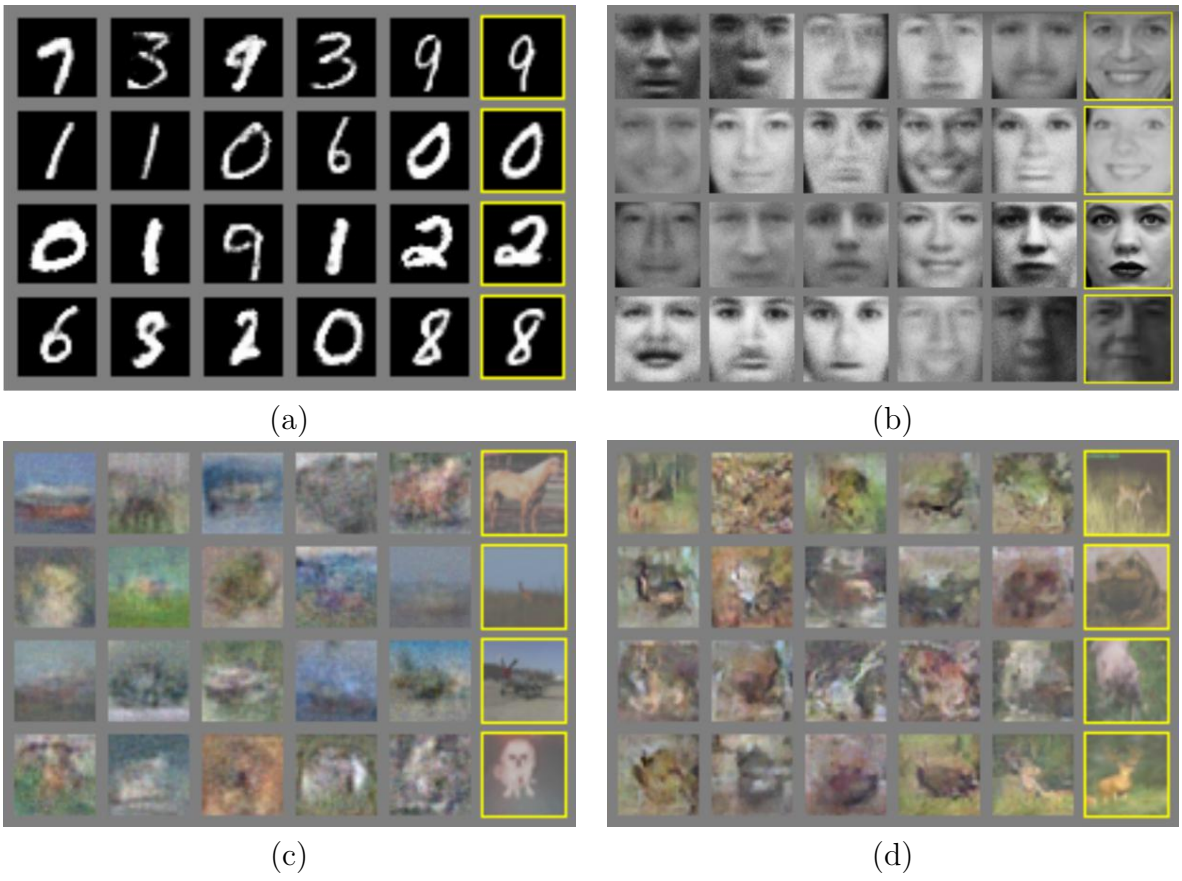


Fig. 2.1 General pipeline for training a generator of images in an adversarial fashion. In training phase, a further network called *discriminator* or *critic* is used to improve the quality of the synthetic images. The two networks are put in competition with each other. The discriminator has to learn useful features to distinguish among real and synthetic (fake) images. At each iteration, the generator tries to fool the discriminator by exploiting the characteristics seen by the discriminator and improving the quality of the synthetic images. In turn, the discriminator evolves and finds more complex features to distinguish among real and fake images. The generator shown in this picture maps random noise into the manifold of images depicting numbers (the MNIST dataset [74]).

has the aim to introduce the base techniques used later in this thesis to generate synthetic data (see chapter 3).

2.1 GANs - Generative Adversarial Networks

To learn the distribution of the generator p_g over the data x, Goodfellow et al. [46] define a prior $p_z(z)$ on the input variables z. Then, they represent a mapping $G(z, \theta_g)$ from the input space toward the desired manifold, here represented from the set of images $x \in X$, where G is a differentiable function represented by a neural network with learnable parameters θ_g. They also define a *discriminator* $D(x, \theta_d)$ as a multi-layer perceptron that outputs a single scalar. In this case, $D(x,)$ represents the probability that x comes from the data rather then from the distribution of the generator p_g. They train D to maximize the probability of assigning the right label to both the training examples x and the synthetic examples generated with $G(z)$. Simultaneously, they train the generator G in such a way that it has to minimize $\log(1 - D(G(z)))$. The loss used in both cases is the log loss. G and D in this context are playing a two-players min-max game with



(a) (b) (c) (d)

Fig. 2.2 Examples of generated images with Generative Adversarial Networks (GANs). To prove that the network does not overfit, in the last column is shown the closest real sample in the feature space. The four versions have been trained to map noise respectively in the (a) MNIST (b) TFD (c) CIFAR-10 - fully connected model (d) CIFAR-10 - convolutional generator/discriminator manifolds.

value function $V(G, D)$:

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{data}(x)} [\log D(x)] + \mathbb{E}_{z \sim p_z(z)} [\log (1 - D(G(z)))] \tag{2.1}$$

This is equivalent of minimizing the Jensen-Shannon divergence which is a smoothed and symmetric version of the Kullback divergence:

$$JSD[P\|Q] = JSD[P\|Q] = \frac{1}{2} KL \left[P \middle\| \frac{P+Q}{2} \right] + \frac{1}{2} KL \left[Q \middle\| \frac{P+Q}{2} \right]. \tag{2.2}$$

where P and Q here represent for shortness the synthetic and the real distribution. Figure 2.2 shows synthetic samples generated with this method, starting from random noise.

The training of a GAN corresponds to find the Nash equilibrium [87] in a non-convex game. Using Gradient descent to seek for a Nash equilibrium may fail: the solution may not converge. Furthermore, the solution oscillates rather than converges to a point. To illustrate the situation with an example, suppose that two agents, namely A and B, have respectively to minimize and maximize the parametric function $f = xy$. The two agents can only manipulate respectively the parameters x and y. The Nash equilibrium in this case occur when $x = y = 0$. When $x = 0$, agent A does not change the value, regardless of what agent B sets as a value of its control parameter y (and viceversa). If $x = 1$, agent A will change the sign of its parameter making it negative to win. The value function is defined as:

$$V(x, y) = xy \tag{2.3}$$

Applying gradient descent, the change in parameter x at each step is:

$$\frac{\partial x}{\partial t} = -\frac{\partial V}{\partial x} = -\frac{\partial xy}{\partial x} = -y(t) \tag{2.4}$$

Similarly, for agent B that operates on paramter y, the update with gradient descent is:

$$\frac{\partial y}{\partial t} = -\frac{\partial V}{\partial y} = -\frac{\partial xy}{\partial y} = x(t) \tag{2.5}$$

where t is the number of the iteration. If we combine the two equations, we get:

$$\frac{\partial y}{\partial t} = x(t) \tag{2.6}$$

$$\frac{\partial^2 y}{\partial t^2} = \frac{\partial x}{\partial t} \tag{2.7}$$

$$\frac{\partial^2 y}{\partial t^2} = -y(t) \tag{2.8}$$

The solution will be:

$$\begin{aligned} x(t) &= x(0)cos(t) + y(0)sin(t) \\ y(t) &= x(0)sin(t) + y(0)cos(t) \end{aligned} \tag{2.9}$$

From this simple example, we can understand two important things: the initial values of the two parameters are foundamental for the quality of the generation and that the system, rather then converging, oscillates.

2.2 DC-GANs - Deep Convolutional Generative Adversarial Networks

Deep Convolutional GANs, introduced by Redford et al. [93], are similar to GANs (section 2.1), with some differences. First of all, both generator and discriminator are fully convolutional, therefore they replaced the fully connected layers present in the GANs. Second, they constraint the output of the generator with a TanH function, because they observed that using a bounded activation allowed the model to learn more quickly to saturate and cover the color space of the training distribution. They also replaced pooling layers with strided convolutions in the discriminator and fractional-strided convolutions in the generator. Finally, they replaced ReLUs in the discriminator with Leaky ReLUs. Figure 2.3 shows the architecture adopted for the generator. As it's possible to see in figure 2.4, these adjustments increase a lot the quality of the generated images, however the DC-GANs are not suitable to generate high resolution images.

2.3 cGANs - Conditional Generative Adversarial Networks

One of the main problems with GANs, is that the task of the generator is a lot harder with respect to the one of the discriminator. What happens sometimes is that, especially at the beginning, the distribution of the generated samples is far from the real samples (figure 2.5). Therefore, there are many sub-optimal discriminators that can separate the two distributions, each one of them unable to give useful information on the appearance and shape of the contents depicted in the real samples. This occurs because, since the discriminator is evaluating unuseful features, the generator will try to enhance those

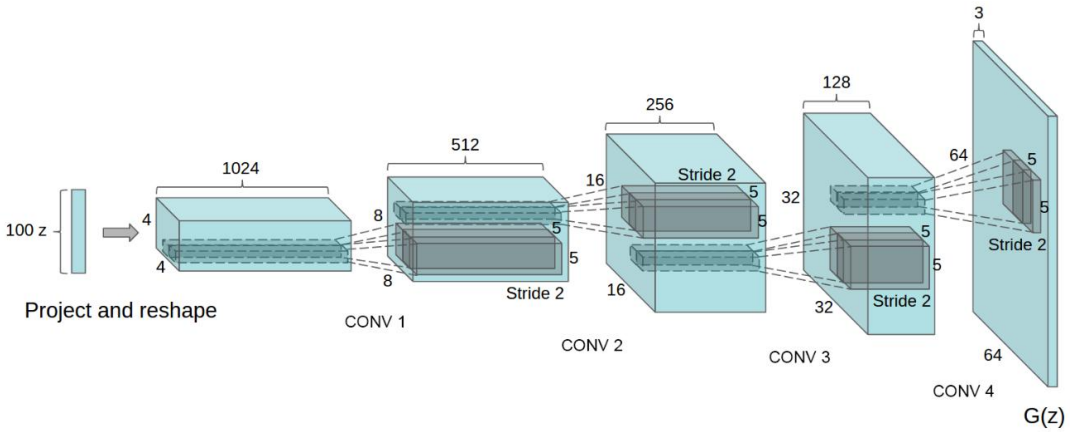


Fig. 2.3 Architecture of the generator network.

Fig. 2.4 Images of bedrooms generated with the DC-GANs [93].

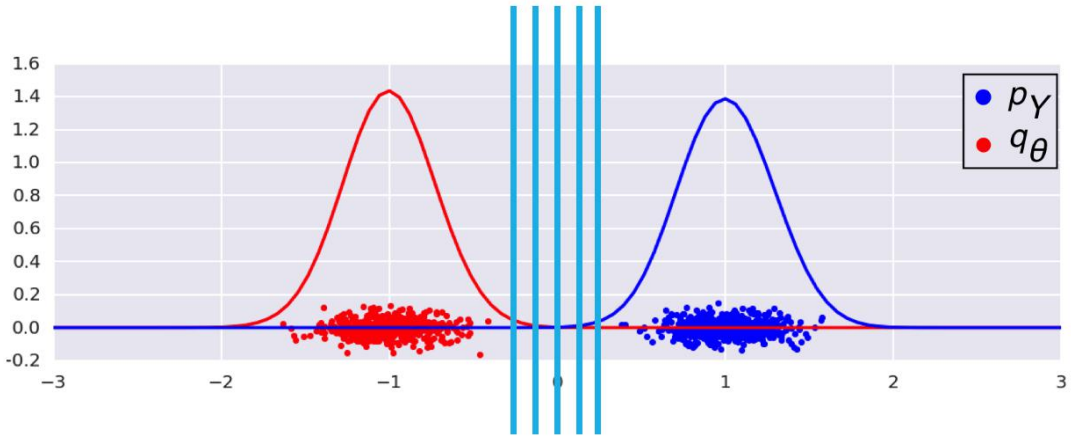


Fig. 2.5 When the distribution of the generated samples is too far from the distribution of the real samples, there are many sub-optimal discriminators (here highlighted in light blue). When this happens, the discriminator is not compelled to learn useful features to discern among the two distributions. Consequently, the generator does not have a beneficial feedback able to increase the quality of the generated output.

features in the synthetic examples. If this happens, the system will not converge. Starting from the analysis of this problem, Mirza et al. [83] improved the stability of the system by introducing a new constraint. Specifically, they force the discriminator to distinguish not only from real or fake examples but also to make a classification task between other classes. For instance, by introducing a new classification task in the discriminator's duty, it has to define and evaluate features also in the real samples. Those features result very useful for the generator since now with this new constraint represent real characteristics of the original manifold that must be considered in order to create engaging synthetic contents. This operation is called *conditioning*. Of course this operation is not always possible to apply, maybe because there aren't classes to distinguish from in the original manifold, or maybe because the ground-truth associated to this new classification task is not available. The *conditioning* solves in part the problems of the DC-GAN, however more steps must be done in order to have an adversarial scheme able to converge in any case.

2.4 wGANs - Wasserstein Generative Adversarial Networks

In the work of Arjovsky et al. [7], they adopt a more general formulation of adversarial training. Specifically, they see it as a training system guided by a distance measure

$d(p_r, p_g)$ between the distributions of the real (p_r) and the generated (p_g) samples. In this conception, instead of using the *log* loss as a distance measure like it happens in GANs (see section 2.1), they investigate different losses such as the total variation (TV) distance and the Jensen-Shannon (JS) divergence and finally they adopt the earth-mover (EM) [76] or Wasserstein distance [97]. Let $\pi(P_r, P_g)$ be the set of all joint distributions γ whose marginal distributions are P_r and P_g. Then:

$$W(P_r, P_g) = \inf_{\gamma \in \pi(P_r, P_g)} \mathbb{E}_{(x,y)\sim\gamma} [\|x - y\|] \tag{2.10}$$

where inf is the *infimum* (also known as *greatest lower bound*).

The earth-mover distance is not so simple to understand. First of all let's talk about the goal of the EM distance. Probability distributions are defined by how much mass they put on each point. Imagine we started with distribution P_r and wanted to move mass around to change the distribution into P_g. Moving mass m by distance d costs $m \cdot d$ effort. The earth mover distance is the minimal effort we need to spend to move one distribution toward the other. So now the question is, why do we need to use the *infimum* to calculate the minimum effort? If we see each $\gamma \in E$ as a transport plan, in order to execute it we need to move $\gamma(x, y)$ mass from x to y for each x,y. Two properties must be respected in order to transform P_r into P_g:

- The amount of mass that leaves x is $\int_y \gamma(x, y)dy$. This must equal the amount of mass originally at x.
- The amount of mass that leaves y is $\int_x \gamma(x, y)dx$. This must equal the amount of mass originally at y.

This shows why the marginals of $\gamma \in \pi$ must be P_r and P_g. For scoring, the effort spent is $\int_x \int_y \gamma(x, y)$. To enforce the Lipschitz constraint [36] on the critic's model, Arjovsky et al. propose to clip the weights.

To better understand the earth-mover distance, let's consider the example shown in figure 2.6. We have 6 boxes of equal unary weight and we want to move them from the left to the right in the placeholders marked by dashed squares. The moving cost of each box is equal to its weight times the distance. As it's possible to see in figure 2.6, the cost of moving the box number 1 from position 1 to position 7 (for a total distance of 6 positions) is $(7 - 1) \times 1 = 6$. Figure 2.7 shows two different moving plans, γ_1 and γ_2. For example, in the first plan γ_1, we move 2 boxes from location 1 to location 10 and the entry $\gamma(1, 10)$ is therefore set to 2. Both plans have a cost of 42. Of course not all plans have the same cost.

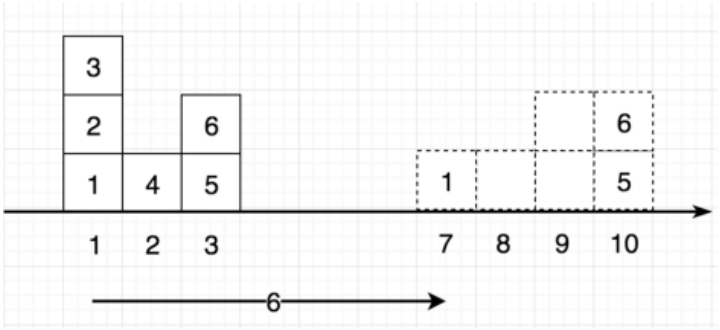


Fig. 2.6 The cost of moving a box of weight w from pos_{start} to pos_{end} is equal to its weight w times the distance $d = pos_{end} - pos_{start}$. Moving block number 1 with unary weight from position 1 to 7 will therefore cost $(7-1) \times 1 = 6$.

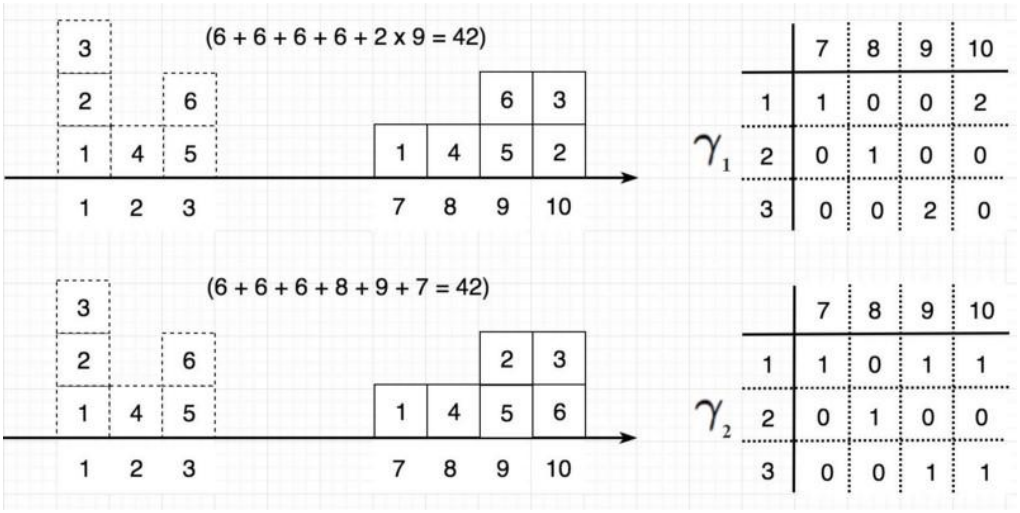


Fig. 2.7 γ_1 and γ_2 are two different plans. Each plan is composed by the number of movements from a starting position to the ending position, for example $\gamma(1, 10) = 2$ means that two boxes have been moved from location 1 to location 10. γ_1 and γ_2 have the same overall cost, however not all possible plans have the same cost.

2.5 wGANs-GP - Wasserstein Generative Adversarial Networks with Gradient Penalty

Gulrajani et al. [48] state that the weight clipping strategy proposed by Arjovsky et al. [7], is not the best way to enforce a Lipschitz constraint. This is because if the clipping parameter is large, it can take a long time for any weight to reach their limit, making it harder to train the critic until optimal. At the contrary, if the clipping is small, this can easily lead to vanishing gradients when the number of layers is high. To overcome this problem, the propose a mechanism called *gradient penalty*. This mechanism, instead of clipping the weights, penalizes the model if the gradient norm moves away from its target normal value of 1. The final loss becomes then:

$$L = \mathbb{E}_{x\sim\mathbb{P}_g}\left[d(\tilde{x})\right] - \mathbb{E}_{x\sim\mathbb{P}_r}\left[d(x)\right] + \lambda\mathbb{E}_{\tilde{x}\sim\mathbb{P}_{\tilde{x}}}\left[\left(\left\|\nabla_{\tilde{x}}D(\tilde{x})\right\|_2 - 1\right)^2\right] \tag{2.11}$$

where:

$$L_{wass} = \mathbb{E}_{x\sim\mathbb{P}_g}\left[d(\tilde{x})\right] - \mathbb{E}_{x\sim\mathbb{P}_r}\left[d(x)\right] \tag{2.12}$$

is the original Wasserstein loss, while the second part:

$$P = \lambda\mathbb{E}_{\tilde{x}\sim\mathbb{P}_{\tilde{x}}}\left[\left(\left\|\nabla_{\tilde{x}}D(\tilde{x})\right\|_2 - 1\right)^2\right] \tag{2.13}$$

is the gradient penality term used to ensure Lipschitzness. Batch normalization is not used in the critic to avoid correlation between samples, and therefore breaking the just-described constraint.

wGANs-GPs represent *de facto* the most stable technique to create accurate generators. In the next chapter I will propose a method for data augmentation that partially exploits this technology to create new synthetic samples.

Part II

Data Augmentation

Chapter 3

Data Generation

"Fantasy is hardly an escape from reality. It's a way of understanding it."

Lloyd Alexander

In this chapter, a novel deep-learning based method for data augmentation is proposed. This method is suited for texturized images and is able to create new synthetic samples for per-pixel classification tasks such as semantic segmentation or anomaly detection. The output is composed by a set of completely new RGB images together with their associated semantic layouts.

The generation process is carried out in three main steps described in Section 3.2: In the first step the system creates new semantic layouts that will represent the generated ground-truth. This step exploits adversarial training (see chapter 2 for further details). In the second step a similarity search is performed among semantic layouts in the input dataset to find similar associated texture as source for the last step. In the third step new images are created by synthesizing new images conditioned on the generated semantic layouts. The new pairs composed by images and layouts can be used to increase existing datasets without any further processing steps.

The effectiveness of the proposed method will be shown by training three architectures for semantic segmentation on the generated synthetic dataset and testing on real images. It will be shown that the networks increase their performances if trained *only* on synthetic images in contrast to the performances obtained with real data. In some cases the relative improvement is above the 90%. Moreover we carried out a comparison with classical data-augmentation approaches. It has been noticed that the proposed data augmentation outperforms its counterpart and can be used together with it in order to further improve models' generalization capabilities. Finally it has been observed that, models trained with

this data augmentation, tend to improve particularly on classes with lower cardinality. This suggests its effectiveness in dealing with the well known long-tail problem.

In Section 3.2 it's disclosed the proposed method in details by exposing every single step from the generation of the semantic layout to the generation of the RGB images. In 3.3 it has been measured the effectiveness of the proposed method by using the generated synthetic images to train different neural network architectures and testing them on real data.

3.1 Related works

Data augmentation was firstly introduced by Krizhevsky et al. [68]. Their work showed how the synthetization of new examples starting from the trainset, helps to reduce overfitting and makes a classification system more accurate in prediction with never-seen data. Specifically, from a real sample, they created a set of new synthetic samples through the application of simple transformations such as translation and horizontal reflection. This work inspired many researchers to explore and exploit data augmentation. In particular, Cubuk et al. [30] defined a simple mechanism called *AutoAugment* of data augmentation that, differently from Krizhevsky et al. [68], which applies the transformations blindly, is able to determine the best data augmentation policy by creating a search space of data augmentation policies and evaluating the quality of each policy as the overall accuracy of the system. A policy consists in the set of transformations introduced by Krizhevsky et al. [68], extended with the operations of rotation and shearing. Furthermore, the frequency of each operation is determined during the search of the best policy. Always in the context of object detection, Schwartz et al. [101] designed a method for integrating a new unseen class in a trained system (operation called *few-shots learning*). This algorithm uses a modified autoencoder, namely a Δ-encoder, both to extract transferable intra-class deformations (Δs) between same-class pairs and to apply those deltas to the few samples of the new introduced class. Buslaev et al. [23] extended the set of transformations and focused their work on increasing the speed of application of the proposed transforms. With the focus moving on data generation through adversarial learning, many researchers adopted generative adversarial networks (GANs) [46] for learning a mapping between a desired input and the manifold of class examples. For example, Antoniou et al. [6] train a generator that learns to modify an existing sample starting from random noise. [9] use a model conditioned on the object images from their marginal distributions to generate a realistic image from their joint distribution by explicitly learning the possible interactions. Relative Appearance Flow Network

(RAFN) is used to generate new viewpoints of a specific object. Zhou et al. [126] learn to synthetize novel viewpoints of an object by predicting the appearance flows. In the context of anomaly detection, Lim et al. [77] use adversarial auto-encoders (AAE) to generate samples from a single class.

3.2 Proposed method

In the current section it is exposed in details this data-augmentation method. The proposed system is composed by three main steps:

- **Semantic layout generation**: A new synthetic semantic layout is generated by means of a Generative Adversarial Network. The network is trained to produce layouts with a similar structure to the original groundtruth layouts of the real set.
- **Image retrieval**: The synthetic layout of the previous step is used to search for similar layouts in the real set. One or more layouts with the corresponding images are selected to be used in the next step.
- **Image generation**: the retrieved layout and the corresponding image are used, together with the synthetic layout, to generate a new image to augment the real dataset.

Figure 3.1 shows the overall pipeline. In the next three subsections we will describe in detail these three steps. As it will be shown in 3.3, the performances of three neural network architectures, trained on synthetized data, is boosted by a large margin. This method is particularly suitable with texturized images, when the amount of data is really low and the dataset distribution has a long-tail form.

3.2.1 Semantic layout generation

The generation of a synthetic semantic layout is achieved by learning a mapping between random noise z and the manifold of semantic layouts $\mathcal{T}$ that compose the ground-truth of the trainset. The synthetization process is entrusted to a convolutional neural network trained through an adversarial training schema. Specifically, it has been adopted Wasserstain GANs with gradient penalty (WGAN-GP) [48], for their stability enforced through gradient penalty and for showing great performances in generating content. The training loss is:

$$L = \mathbb{E}_{x\sim\mathbb{P}_g}\left[d(\tilde{x})\right] - \mathbb{E}_{x\sim\mathbb{P}_r}\left[d(x)\right] + \lambda\mathbb{E}_{\tilde{x}\sim\mathbb{P}_{\tilde{x}}}\left[\left(\left\|\nabla_{\tilde{x}}D(\tilde{x})\right\|_2 - 1\right)^2\right] \tag{3.1}$$

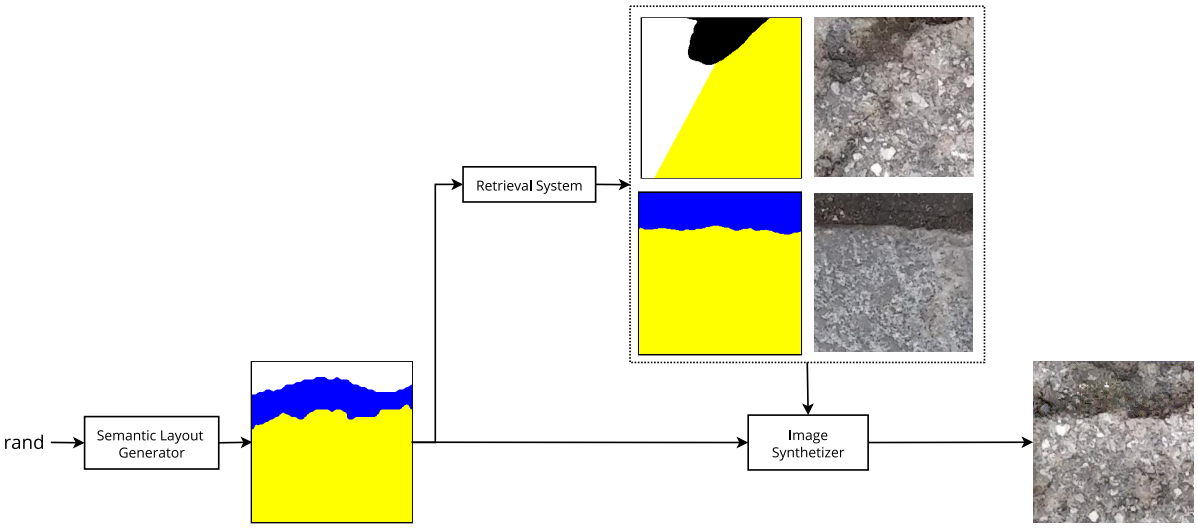


Fig. 3.1 Pipeline of the proposed method.

The network of the generator is inspired by [48] and is composed by a first layer that maps a noise-map of size 4×4 in a set of $16N$ activation maps of the same size, where N is the number of filters desired in the last layer. Afterwards, a set of 4 blocks iteratively upsamples the featuremaps to reach the final size of (64×64). Each block is composed by three sub-blocks: the first increases the number of feature-maps as a preprocessing for sub-pixel upsampling and is composed by a convolution layer with kernel size 3×3 and padding 1, batch normalization and ReLU. Afterwards, the featuremaps are upsampled of a factor 2 using PixelShuffle [102] layer. The block terminates with a final processing composed by a convolution of 3×3 with padding 1, a batch normalization and a ReLU. Each $i-th$ block transforms $2^{4-i+1}N$ feature maps into a set of $2^{4-i}N$ and extends the spatial extend of a factor 2. Table 3.1 shows in detail the composition of the generator.

Similarly, the critic used in this step is composed by a set of downsizing blocks composed by a convolutive layer of kernel 3×3 and padding 1 followed by batch normalization and a max pooling of size and stride 2. A final convolution with kernel size 4, projects the last featurmap having size $8N \times 4 \times 4$ into a single number, that will be the output of the critic. Table 3.2 shows in detail the composition of the critic. For both networks, the batch normalization has a momentum of 0.01. The output is upsampled with a nearest neighbour strategy to a scaling factor 4, reaching a spatial dimension of (224×224). The generation of a smaller output and the later upsampling, allows us to have a system able to work on very small datasets. This technique cannot be used on RGB images, since the scaling of the pixels would lead to blurry images that would not present small local structures.

Table 3.1 Structure of the convolutional neural network of the generator. All convolutional layers have a kernel size of 3 and a padding of 1, except for the last one that has a kernel size of 1 and stride 1.

Stage	Operation	Output size
Pre-processing	Input	$1 \times 4 \times 4$
	Conv + B.N. + ReLU	$16N \times 4 \times 4$
Conv. Network	Conv + B.N. + ReLU	$64N \times 4 \times 4$
	PixelShuffle	$16N \times 8 \times 8$
	Conv + B.N. + ReLU	$8N \times 8 \times 8$
	Conv + B.N. + ReLU	$32N \times 8 \times 8$
	PixelShuffle	$8N \times 16 \times 16$
	Conv + B.N. + ReLU	$4N \times 16 \times 16$
	Conv + B.N. + ReLU	$16N \times 16 \times 16$
	PixelShuffle	$4N \times 32 \times 32$
	Conv + B.N. + ReLU	$2N \times 32 \times 32$
	Conv + B.N. + ReLU	$8N \times 32 \times 32$
	PixelShuffle	$2N \times 64 \times 64$
	Conv + B.N. + ReLU	$N \times 64 \times 64$
Post-processing	Conv	$C \times 64 \times 64$

Table 3.2 Structure of the convolutional neural network of the critic. All convolutional layers have a kernel size of 3 and a padding of 1, except for the last one that has a kernel size of 4 and stride 1.

Stage	Operation	Output size
Pre-processing	Input	$C \times 64 \times 64$
Conv. Network	Conv + ReLU + MaxPool	$N \times 32 \times 32$
	Conv + ReLU + MaxPool	$2N \times 16 \times 16$
	Conv + ReLU + MaxPool	$4N \times 8 \times 8$
	Conv + ReLU + MaxPool	$8N \times 4 \times 4$
Post-processing	Conv	$1 \times 1 \times 1$
	View	1

3.2.2 Image retrieval

It has been observed that the quality of the final RGB image synthesized in the last step depends heavily on the choice of the samples of the trainset used for the generation. As it's described later in fact, for rendering a semantic portion as realistic RGB content, it's required to pick a sample from the trainset containing the same semantic content. It has been noticed that the similar is the size of this real semantic portion with respect to the synthetic semantic zone, the better will be the results in the RGB generation process. Furthermore, since there is a visual consistency by adjacent semantic portions, the more classes are matched in one sample, the more realistic will appear the final image. Two metrics compose the *consistency score*, the *intersection*, which represents the number of common semantic classes between the synthetic S_s and the real S_t sample, and the *coverage factor*, which indicates the average ratio between the two corresponding areas. Let H_s and H_t be respectively the histograms of S_s and S_t representing the number of pixels for each semantic zone. The *intersection* is defined as:

$$intersection(S_s, S_t) = \sum_{i=0}^{C} [(H_s(i) > 0)(H_t(i) > 0)] \tag{3.2}$$

where C is the number of classes. The *coverage factor* is defined as:

$$converagefactor = \frac{H_s}{H_t} \tag{3.3}$$

Defined $\mathcal{I}$ as the set of real samples having the maximum intersection with S_s, the final sample S_t^* that will be associated to the synthetic map will be chosen from the subset of samples having the maximum intersection:

$$S_t^* =_{S_i \in I} [intersection(S_s, S_i)] \tag{3.4}$$

This process is repeated until all classes have been matched with a ground-truth sample. At the end of each iteration, the pixels corresponding to the matched classes are zeroed and the process is repeated. The output of this operation, is a set ground-truth samples and, for each sample, the classes that will be rendered from that sample. For example, it's possible to see in figure 3.1 the output of this operation. To the generated semantic layout are associated two real semantic layouts. The first one is used to match the white and the yellow region, while the second one only to generate the blue portion. Algorithm 1 defines more in detail the just-described image retrieval system.

Algorithm 1: Procedure for finding ground-truths in the trainset with similar areas with respect to the generated one.

Input: $H \times W$ gen. sem. image $S_s \in \{0, \dots, C-1\}$
Input: ground-truths $\mathbb{T}$
Input: min coverage allowed p_{min} in perc.
Input: max coverage allowed p_{max} in perc.
Output: Collection of similar layouts $\mathcal{S}$
Output: Collection $\mathcal{F}$ of class ids to get from each layout in $\mathcal{S}$
$\mathcal{S} \leftarrow []$;
$\mathcal{F} \leftarrow []$;
while *true* **do**
  $intersect_{best} \leftarrow 0$;
  $coveragefactor_{best} \leftarrow 0$;
  $S_t^{best} \leftarrow null$;
  $\mathbb{T} \leftarrow$ random sort $\mathbb{T}$;
  $H_s \leftarrow$ class histogram of S_s;
  for S_t *in* $\mathbb{T}$ **do**
    $H_t \leftarrow$ class histogram of S_t;
    $coveragefactor_{cur} \leftarrow H_s / H_t$;
    $intersect_{cur} \leftarrow \sum_{i=0}^{C} [(H_s(i) > 0)(H_t(i) > 0)]$;
    if $intersect_{cur} >= intersect_{best}$ **then**
      if $coveragefactor_{cur} >= coveragefactor_{best}$ **then**
        $intersect_{best} \leftarrow intersect_{cur}$;
        $coveragefactor_{best} \leftarrow coveragefactor_{cur}$;
        $S_t^{best} = S_t$;
      end
    end
  end
  $\mathcal{S} \leftarrow \mathcal{S} \cup S_t^{best}$;
  $H_t^{best} \leftarrow$ class histogram of S_t^{best};
  foundclasses $\leftarrow argwhere\left[(H_s > 0)(H_t^{best} > 0)\right]$;
  $\mathcal{F} \leftarrow \mathcal{F} \cup foundclasses$;
  for c *in foundclasses* **do**
    $S_s[S_s = c] \leftarrow 0$
  end
  if $\sum_r \sum_c S_s(r, c) = 0$ **then**
    break
  end
end
return $\mathcal{S}$,$\mathcal{F}$

3.2.3 Image generation

This part of the pipeline concerns the production of new synthetic images. First, a brief recap of the overall method: the first step consists in generating a new synthetic layout as presented in Section 3.2.3. As a second step the synthetic layout is used to retrieve a similar layout from the real set together with the associated real image. Finally the third step, described in the current section, consists in the generation of a new synthetic image conditioned on the synthetic layout. Our image generation method produces, in the majority of cases, synthetic samples that are almost indistinguishable from the real ones. Despite that, for our application, we actually do not care if the images are perceptually similar to the real one as long as the model benefits from training on the synthetized data.

Texture Synthesis using CNNs: Gatys et al. presented in [42] a work on texture synthesis using Convolutional Neural Networks. The idea is to use a pretrained Neural Network to shape a random initialized image to match the feature distribution of a real input image. Given a source texture image, first features are extracted from this image, then a summary statistics is computed on the features to obtain a stationary description of the source image. Finally, a white noise image is shaped to have the same stationary description by performing gradient descent from the feature extractors layers to the white noise image.

Our method for texture synthesis: It is heavily inspired by [42] but we enhanced it by adding the capability to synthesize different textures from a single image and even from multiple images. As in [42] we characterize a texture by passing it through a pretrained convolutional neural network (i.e. the VGG-19 model [103]). We obtain a list of activation maps for each layer in the network and store each of them in a matrix $F^l \in \mathbb{R}^{\mathbb{N}_l \times \mathbb{N}_l}$ where F^l_{jk} is the activation of the j^{th} filter at position k in layer l. Then we compute a masked Gram matrix $G^{lc} \in \mathbb{R}^{\mathbb{N}_l \times \mathbb{N}_l}$ on each F^l to obtain a stationary summary discarding spatial information. This representation will be used to generate the synthetic image. The Gram matrix G^{lc}_{ij} is computed as

$$G^{lc}_{ij} = \sum_k F^l_{ik}\delta(M_{ik} = c)F^l_{jk}\delta(M_{jk} = c) \tag{3.5}$$

where G^{lc}_{ij} is the inner product between the feature maps i and j in layer l and class c. M is a spatial mask that codifies information about the spatial layout of the texture. In each position M_{ik} it is stored the class index (in our case the texture type). Then $\delta(\cdot)$ is a simple dirac function that output 1 in correspondence to the correct class index and 0 otherwise. A Gram matrix G^{lc}_{ij} is computed for each class in the layout mask and for

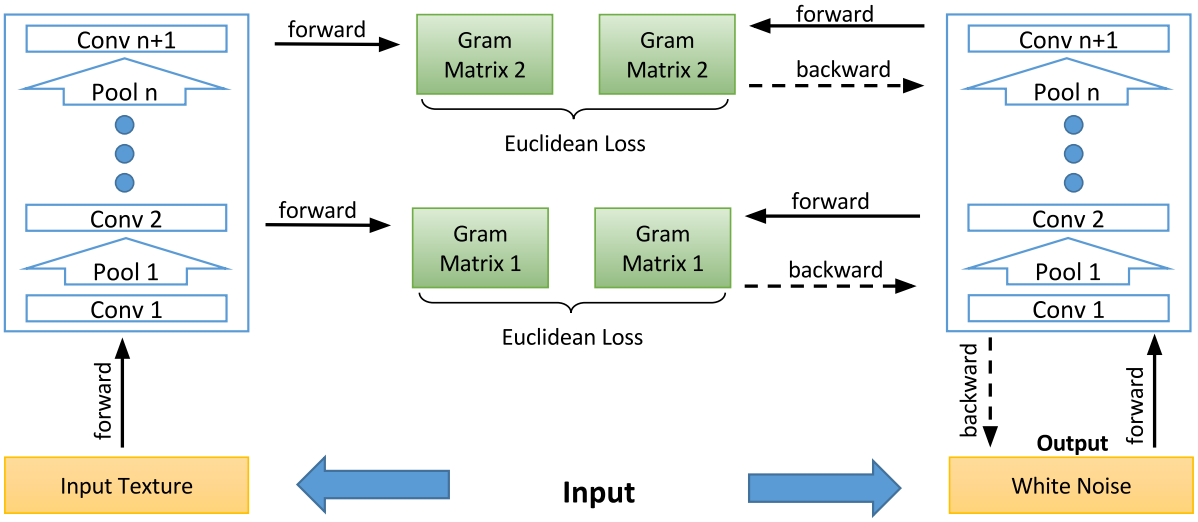


Fig. 3.2 Texture synthesis using CNNs. White noise image is modified by backpropagation in order to obtain the synthetic image.

each network activation layer we intend to use in the optimization process. The new synthetic image is generated by gradient-based optimization. We initialize the new image with white noise. The loss function is a mean-squared distance between the Gram matrix of the real image and the Gram matrix of the image being generated. The partial loss L_{lc} for a specific layer l and a specific class c can be expressed as:

$$L_{lc} = \frac{1}{4N_{lc}^2 M_{lc}^2} \sum_{i,j} (G_{ij}^{lc} - \hat{G}_{ij}^{lc})^2 \tag{3.6}$$

and finally the total loss between the real image $\vec{x}$ and the image being generated $\hat{\vec{x}}$ is expressed as:

$$\mathcal{L}(\vec{x}, \hat{\vec{x}}) = \sum_{l=0}^{L} \sum_{c=0}^{C} w_{lc} L_{lc} \tag{3.7}$$

where w_l are weighting factors for each layer and each class. Notice that, since we optimize only the part of the target image with a specific class index, we can optimize using different source images containing different classes. Thus in our image generation process, after retrieving the image with the largest *coverage factor* and optimizing the corresponding classes, if there are some classes in the target image not yet optimized, we run the optimization process twice masking out the image regions already optimized. Figure 3.2 shows in details this generation process.

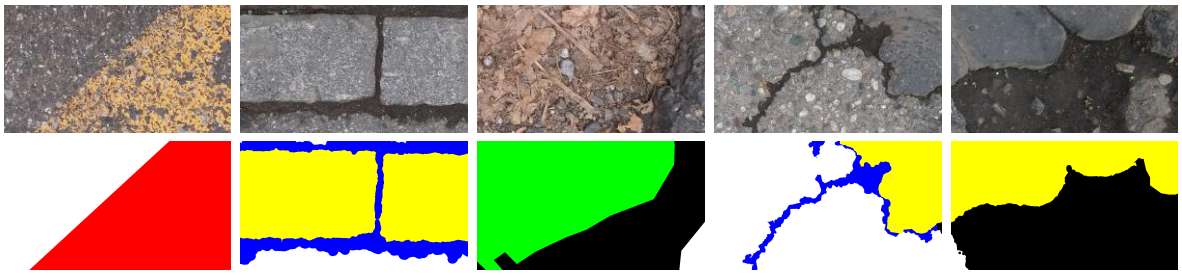


Fig. 3.3 Random samples thrown from the dataset collected and used in this work.

3.3 Experiments

3.3.1 Dataset

The dataset used in this work is composed by 448 images of asphalt acquired in the city of Milan (Italy) and ground-truthed by the author. The dataset is splitted in 314 images of train and 134 images of test. Each sample is composed by an RGB image of size 585×1040 and a corresponding semantic layout that defines, for each pixel, to which semantic class it belongs. Figure 3.3 shows some examples of the collected dataset. The semantic classes here considered are 8: *asphalt*, *lane mark*, *raw asphalt*, *crack*, *dirt*, *terrain* and *gravel*.

3.3.2 Evaluation

To evaluate the effectiveness of our data augmentation technique we trained three different neural networks. For the purpose of this work we are not interested in absolute performance values. Instead we want to assess the relative improvements that our data agumentation technique can bring compared to training models on real data. We chose three quite efficient network architectures because they can be trained and evaluated very fast. Moreover we observed significant differences in performances due to different random initialization of network weights. Thus, for each experiment in this work, we trained/tested at least five times each network to improve the statistical significance of our results. The use of efficient architectures allowed us to carry on a feasible set of experiments with reasonable computational time.

ENet [89] is one of the most efficient architectures for semantic segmentation. It adopts an encoder-decoder design with 28 stacked bottleneck layers. It's efficiency is due to different factors, mainly the use of asymmetric convolutions. Different works investigated the use of asymmetric convolutions both from a theoretical point-of-view [105, 3] and in a practical setting [109, 107]. The main idea is to spatially decompose

3×3 convolutional filters into 3×1 followed by 1×3 filters. The resulting block is 33% more efficient in terms of FLOPs and allows to insert a further non-linearity in between the convolutions. This theoretically increases the representational power of the whole convolution block [3].

ERFNet [95] can be located in the middle between an efficiency oriented architecture and an accuracy oriented one. It adopts an encoder-decoder structure inspired by ENet where the main building block is modified to increase model accuracy. It employs a new block called Non-Bottleneck-1D and a fast downsampling strategy where the inner activations are spatially downsampled in the earlier part of the network to keep the majority of the computational burden for the latest stages.

GUNet [82] adopts a multi-resolution encoder to exploit at the same time low-resolution structures and high-resolution details. The two resolutions are processed by different branches of the network which shares the same weights. Multi-resolution features are fused by means of a Fusion Module. The decoder part consists of a Guided Upsampling Module to efficiently output the prediction map at full resolution.

3.3.3 Results

To show the effectiveness of our data augmentation method, we trained the three different network architectures described in subsection 3.3.2 on different types of data (real and/or synthetic), with our without geometrical data augmentation. Two experiments have been done. The first one has the aim to study the increase of the performances by using *only* synthetic samples, in function of the number of used synthetic samples. As it's possible to observe from figure 3.4, less than 1300 synthetic samples are required to have a system trained only on generated data that outperforms the same system trained only on real data. To make this experiment statistically relevant, each point in the graph is the average of the performance scores obtained in 10 runs.

The second experiment shows the performances achieved by the three semantic segmentation architectures using real and/or synthetic data proposed by our method with or without geometrical data augmentation. This experiment shows clearly that the use of synthetic data significantly improves the performances of the three systems. This gain in performances is a lot more accentuated on classes with low-cardinality, such as *Dirt*, *Terrain* and *Gravel*. These classes in fact, score 0 if the system is trained with real data only.

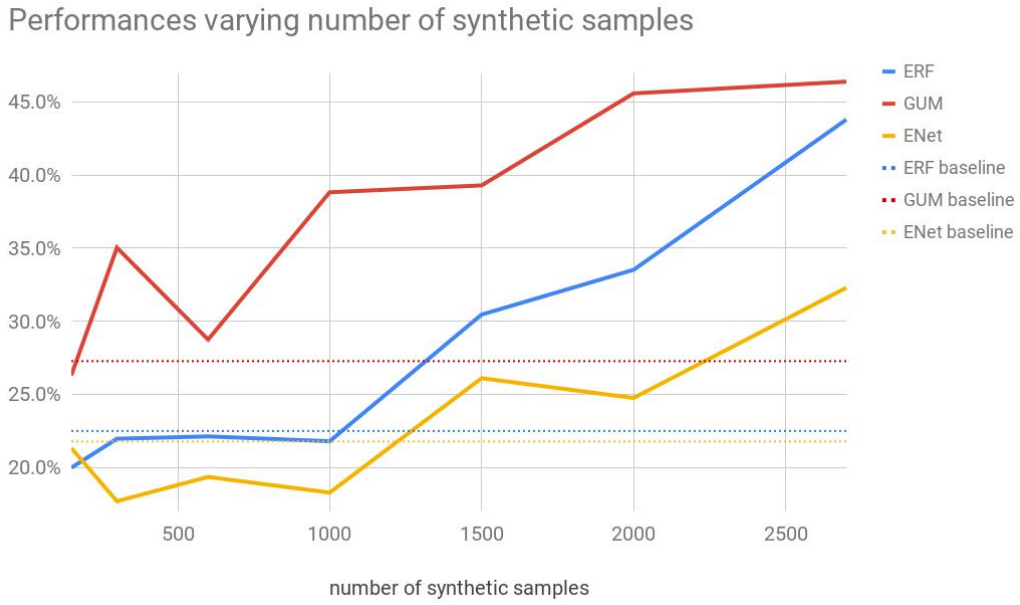


Fig. 3.4 Increase of the performances varying the number of synthetic samples. In this case the systems never saw real data during training. As it's possible to observe, with less then 1300 synthetic samples, all the methods outperform the results achieved with real data only.

	R	R+DA	S	S+DA	R+S	R+S+DA
Asphalt	63.2	71.8	47.0	66.1	53.2	55.3
Lane mark	79.2	86.1	90.0	85.0	82.9	81.8
Raw asphalt	0.0	0.0	14.0	22.0	21.1	17.7
Crack	0.1	0.0	17.7	21.7	17.0	22.2
Dirt	2.4	0.3	30.9	21.8	47.8	56.9
Terrain	0.1	0.0	12.4	18.4	8.9	14.5
Gravel	13.2	5.5	18.3	26.2	21.9	30.7
AVG	**22.6**	**23.4**	**32.9**	**37.3**	**36.1**	**39.9**

(a) GUNet

	R	R+DA	S	S+DA	R+S	R+S+DA
Asphalt	69.8	66.2	49.2	62.1	60.9	55.6
Lane mark	84.6	84.5	76.8	75.5	77.9	90.3
Raw asphalt	0.0	0.0	14.4	15.9	22.6	30.0
Crack	0.0	0.0	16.8	21.4	20.7	29.8
Dirt	0.1	0.0	29.0	18.6	37.7	33.9
Terrain	0.0	0.0	7.0	14.5	10.4	18.6
Gravel	9.8	9.9	18.7	26.6	24.3	28.9
AVG	**23.5**	**22.9**	**30.3**	**33.5**	**36.4**	**41.0**

(b) ENet

	R	R+DA	S	S+DA	R+S	R+S+DA
Asphalt	67.1	70.2	55.2	57.1	39.1	55.4
Lane mark	83.3	84.5	85.6	84.2	81.0	79.6
Raw asphalt	0.0	0.0	18.3	19.7	12.7	21.8
Crack	0.0	0.1	18.6	25.8	18.1	27.0
Dirt	0.1	3.9	26.1	50.3	34.3	47.2
Terrain	0.0	0.0	9.7	19.7	18.9	17.4
Gravel	16.8	7.9	18.0	21.4	22.5	24.1
AVG	**23.9**	**23.8**	**33.1**	**39.8**	**32.4**	**38.9**

(c) ERFNet

Fig. 3.5 Performances achieved by the three semantic segmentation architectures using real (R) and/or synthetic data (S) proposed by our method with or without geometrical data augmentation (DA). As it's possible to see, all networks have a increase of the performances when trained also with synthetic samples, which is highly accentuated in long-tail classes such as *Dirt*, *Terrain* and *Gravel*, where the use of only real data is not sufficient.

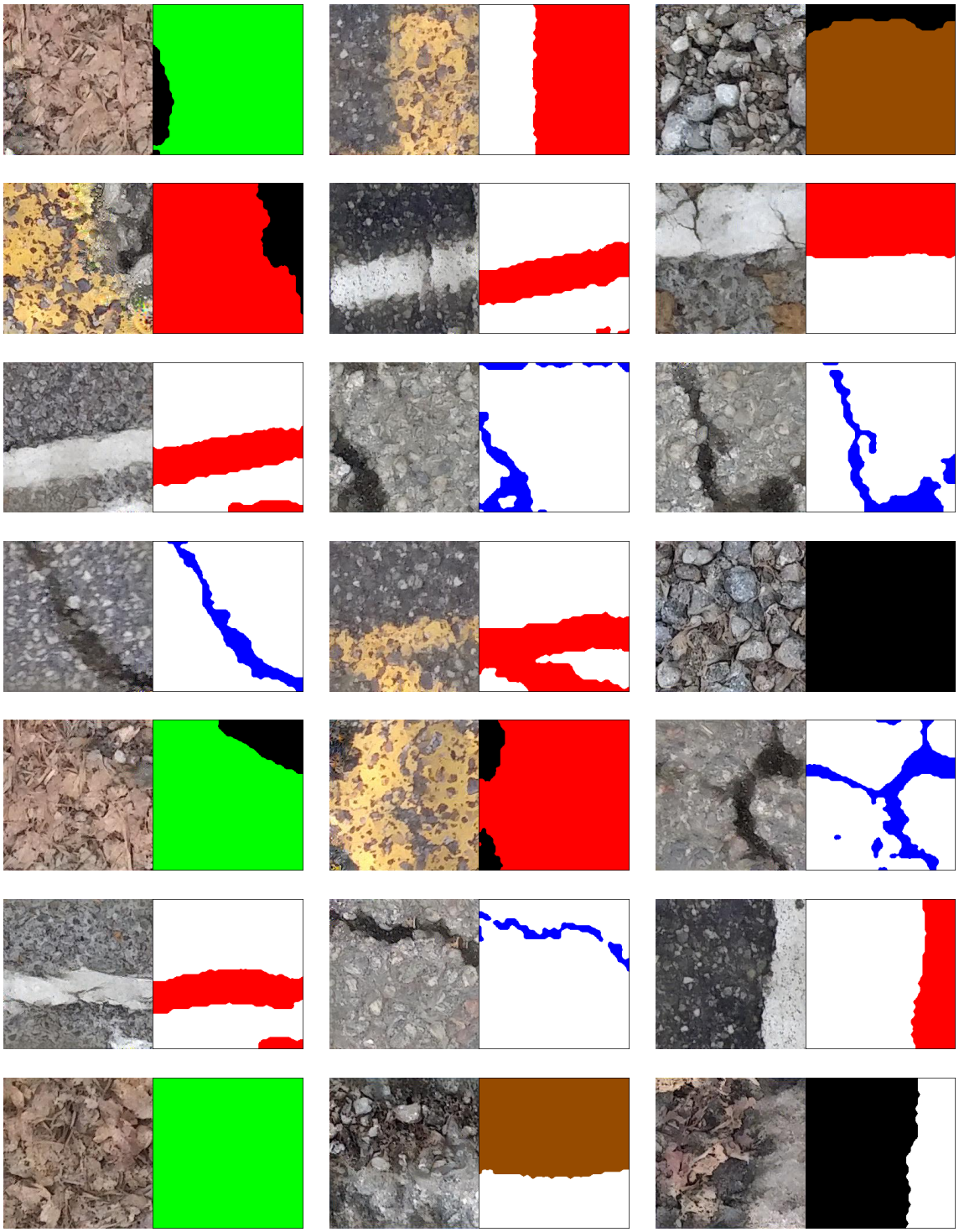


Fig. 3.6 Random synthetic samples.

Part III

Image Enhancement

Chapter 4

Image Enhancement

"Natural beauty takes at least two hours in front of a mirror."

Pamela Anderson

4.1 Introduction

Image enhancement is a foundamental step for anomaly detection. Input images in fact, can be affected by undesired light effects or present very poor contrast. All those factors can decrease the performances of the system, either by creating a huge amount of false positives that will be sent in the discard line or by making anomalies less visible and thus, harder to detect. In the context of this chapter, we address the task of image enhancement on photos depicting *man-made* subjects like the products under inspection as well as on other types of subjects, such as natural landscapes. This relaxation allows us to compare the proposed method with the relative state of the art. In this conception, we present an automatic retouching system that can rival with the ability of an expert human retoucher.

The study of image enhancement algorithms has a long tradition in the field of image processing and a large variety of method have been proposed in the literature [115]. Recently, the problem has been tackled as an application of machine learning. In its conceptually simplest form, the problem is formulated as an *image-to-image translation* [59] between the raw input images and their versions post-processed by expert photographers. This approach led to remarkable results, but it is not computationally feasible to apply it to the high resolution images produced by modern acquisition devices. Image-to-image translation systems are usually run on low resolution thumbnails. Since these systems

need to learn to reproduce the whole content of the images from scratch,they often produce photographs with some visible artifacts or some loss of the original details.

As noticed by Gharbi *et al.* the limitations of image-to-image translation can be avoided if, instead of estimating pixelwise the whole retouched image, only the transformation between raw and retouched images is computed [43]. In this way it is possible to perform the estimation on a small resolution thumbnail and to apply the resulting transformation on the original high-resolution photograph. Moreover, fine image details can be preserved and image artifacts can be avoided by just restricting the transformation to belong to a suitable family of transformations.

In this preprocessing stage it has been followed the latter approach. It has been designed *SpliNet*, a method based on a convolutional neural network (CNN) that analyses raw image to select an image transformation. The transformations considered here are global channel-by-channel transfer functions that resemble those used by the "color curves" feature provided by most photo-editing programs. Color curves are a powerful tool that allows expert retouchers to obtain a variety of effects including brightness and contrast adjustment, color balancing, gamma correction, and many others. The curves are typically defined as the interpolation of a set of reference points that the retoucher places on a plot. Once defined, the curves are globally applied to the pixels of the input image. The curves produced by the proposed neural network are natural cubic splines. Splines have been chosen because they can effectively approximate arbitrarily complex functions.

Moreover, splines are easy to interpret and they would make it possible to design a semi-automatic system, where the automatically estimated color curves could be manually refined by the photographer.

For an automated photo editing system, being able to learn to reproduce the retouching style of a single photographer often is not enough. In fact, there is not a single optimal way of retouching photographs. Each person may prefer a different level of saturation, brightness, contrast, etc. and these preferences may vary with the semantic content of the images. On the basis of these considerations, we extended the neural network to make it able to reproduce the different styles of the users with a single model.

As a further extension, we devised two different strategies that allow,without retraining the network, to model new users on the basis of their retouching preferences on a very small set of images.

To verify the effectiveness of our approach we used the MIT-Adobe FiveK dataset [24]. This dataset consists of 5000 photographs, each one retouched by five photo-editing

experts. The results we obtained demonstrate that the proposed method allows to accurately model both single and multiple users.

4.2 Related work

Image enhancement is a classic problem in image processing which consists in altering an input image to improve its quality. The literature on image enhancement presents three main approaches: *interactive enhancement* methods [78, 5] exploit the interaction with the users to help them in modifying the images to meet their needs; *learning enhancement* methods [63] model the users' preferences and use them to guide the processing steps; *automatic enhancement* methods model the whole enhancement procedure, usually with the help of a training set of reference images that the system learns to reproduce [43].

In this work we focus on automatic enhancement which, over the other approaches, has the advantage of being applicable in a broad set of scenarios including those where the users lack any skill in photo editing or image processing. Methods of this kind can be naturally divided in two categories, namely the *paired* [59] and the *unpaired* [127] ones. The former uses a set of input/output pairs and learns how to reproduce the relationship between them. Methods in the *unpaired* category, instead, use two unrelated sets of input and output images. Learning from unpaired images is clearly more difficult, since it requires to abstract the concept of *enhanced image* and to capture the elements that make an image pleasant to the human observer.

The semantic content can significantly influence the perceived quality of images. To take this into account some methods, that we call *semantic aware*, exploit additional information extracted by specialized detectors. Among potentially valuable cues there are semantic segmentation [79], saliency maps [122], illumination maps [49] etc.

Another important aspect of enhancement methods is the performance criteria which guided their design. Some of them are *accuracy-oriented* and have the main goal of achieving a small difference between the actual and the desired output images; other methods are *speed-oriented* because they have been designed to reduce the processing time; there are also versatile methods that can be tuned in one or the opposite direction by modifying a suitable parameter [15].

The literature on image enhancement is very rich and includes a large variety of methods. Table 4.1 lists a selection of recent methods characterized in terms of the criteria described above. In the following we will describe those that are especially relevant for our work.

Table 4.1 Image enhancement methods in the state of the art. Each method is categorized according to its approach (*interactive*, *learning-based* or *automatic*), the modeling strategy (with *paired* or *unpaired* training samples), its use of additional information (semantic *aware* or *unaware*), and the main performance goal (optimized for *accuracy*, *speed* or *tunable*).

Method	**Enhanc.**			**Info**		**Sem.**		**Goal**		
	interactive	*learning*	*automatic*	*paired*	*unpaired*	*aware*	*unaware*	*accuracy*	*speed*	*tunable*
Bae et al. [10]			✓	✓			✓	✓		
Cohen-Or et al. [28]			✓		✓		✓	✓		
Lischinski et al. [78]	✓				✓		✓	✓		
An and Pellacini [5]	✓			✓			✓	✓		
Kang et al. [63]		✓		✓			✓	✓		
Kaufman et al. [65]			✓	✓		✓		✓		
Bianco et al. [16]			✓	✓		✓		✓		
Yan et al. [121]			✓	✓		✓		✓		
Bianco et al. [17]			✓	✓		✓		✓		
Bianco et al. [15]			✓	✓			✓			✓
Gharbi et al. [43]			✓	✓			✓		✓	
Guo et al. [49]			✓	✓		✓		✓		
Isola et al. [59]			✓	✓			✓	✓		
Zhu et al. [127]			✓		✓		✓	✓		
Bianco et al. [13]			✓	✓		✓		✓		
Hu et al. [54]			✓		✓		✓	✓		
SpliNet (single user)			✓	✓			✓	✓		
SpliNet (multi user)		✓		✓			✓	✓		

Gharbi *et al.* presented a supervised method for image enhancement that focuses on computational efficiency to achieve real-time processing on smartphones [43]. The workload is split in a low-resolution stream and a high-resolution stream. The low-resolution stream, which carries the highest payload, is composed of a convolutional neural network with two terminal branches that respectively estimate local and global features. The outputs of the two branches are later combined together. In turn, the high-resolution stream learns a guidance map used to upsample the features inferred in the low-resolution stream and finally those per-pixel transformations are applied to the input at its full-resolution.

To allow a more precise local enhancement, Yan *et al.* introduced an image descriptor that accounts for the local semantics of the input image [121]. This method applies a color transformation to each pixel of the input image. Specifically, the color transformation used in this case is a polynomial of degree two, therefore each input pixel must be firstly projected into a monomial basis containing the 10 terms of the polynomial expansion. To improve accuracy from a perceptual point of view, the input image is firstly converted in the *CIELab* color space.

Bianco *et al.*, relax the constraint of inferring a per-pixel transform by capturing local and global variations with a set of per-patch color transformations [15]. The dimension of the patch is a parameter that adjusts the algorithm toward accuracy of reconstruction or toward speed, depending on the needs of the user. The smaller the patch size, the higher the accuracy but also the processing time. Artifacts are prevented by constraining to a smooth transition between adjacent transformations. Here, the color transformation is a polynomial of degree three whose parameters are estimated by a neural network.

In contrast to previous methods, Hu *et al.* apply a sequence of simple image manipulations. An adversarial training scheme allows training from unpaired image data [54]. Reinforcement learning is used to make the system able to iteratively choose which operation (if any) need to be applied.

The method proposed by Isola *et al.* also uses adversarial training to perform an image-to-image translation [59]. In this case, the output of the neural network is the final enhanced image. The neural network acts as a regressor, and the price to pay to have such a powerful regressor is that the number of parameters is very high and, therefore, is also high the number of training examples required to make it work properly. Furthermore, since there are no constraints that explicitly preserve the content of the images, this method can introduce undesired local patterns that corrupt the structure of the input image.

The method proposed by Zhu *et al.* exploits cycle redundancy to constrain the type of transformation between a source manifold and a target manifold [127]. Given two unpaired sets of images two convolutional networks are trained to transform images of one kind into images of the other, and vice-versa. Adversarial losses are used to make sure that the concatenation of the two transformations produces consistent results. The approach is similar to that of Isola *et al.* but, since the unpaired training is more challenging, the results it obtains are usually worse.

4.3 Proposed method

In this work we propose a novel CNN-based method that estimates a global color transform for the enhancement of raw images. This method is designed to improve the perceived quality of the images by reproducing the ability of an expert in the field of photo editing. The transformation that is applied to the input image is found by a convolutional neural network that has been specifically trained for this purpose. More precisely, the network takes as input a raw image and produces as output three sets of *control points* (also called *nodes*), one set for each color channel. Then, the control points are interpolated and the resulting functions are globally applied to the values of the input pixels.

As interpolating functions we chose to use natural cubic splines, since they can approximate arbitrarily complex functions (provided that enough nodes are specified) while remaining as smooth as possible (among all interpolating functions natural splines are those with minimal average curvature). The smoothness of splines introduces an implicit form of regularization that prevents the abrupt changes and the artifacts that can be often observed for methods based on look-up tables or on image-to-image translation. An important property of this approach is that, being both smooth and global, the transformation preserve the content of the input images. Moreover, as we will show, spline interpolation can be computed very effectively within a deep learning framework, allowing for a rapid training of the neural network.

The proposed network architecture is end-to-end trainable. This is possible because splines are continuous and differentiable functions. To speed up the computation and to make the method suitable for real-time processing, the estimation of the nodes of the spline is calculated on a downsampled version of the input image of size 256×256, and the enhancement is applied to the original image at full resolution.

We called the proposed method SpliNet; its overall architecture is shown in Figure 4.1. The processing pipeline consists of three major steps: the estimation of the nodes, their

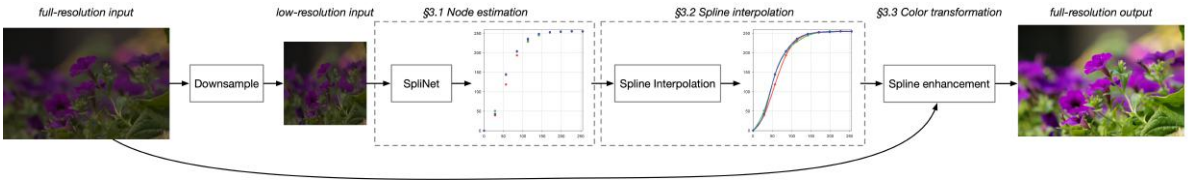


Fig. 4.1 Processing pipeline of the SpliNet image enhancement method. The input image is resized to 256×256. Then the nodes of the spline are computed through a convolutional neural network. Finally, for each color channel a natural cubic spline interpolates the nodes found and the resulting color transformation is applied to the pixels of the input image.

interpolation with splines and, finally, the color transformation of each color channel with the corresponding spline. The next sections describe these steps in detail.

4.3.1 Node estimation

Splines are parametric, piecewise polynomial curves widely used for data interpolation [34]. In this work we considered natural splines that are formed by a set of cubic polynomials. A spline $s(x)$ that is used to interpolate a set of N nodes $(x_1, y_1), \ldots, (x_N, y_N)$ is defined as the combination of $N-1$ polynomials $p_1, p_2, \ldots, p_{N-1}$:

$$s(x) = \left(\sum_{i=1}^{N-2} p_i(x) \chi_{[x_i, x_{i+1})}(x) \right) + p_{N-1}(x) \chi_{[x_{N-1}, x_N]}(x), \tag{4.1}$$

where $[x_i, x_{i+1})$, is the interval assigned to the polynomial p_i, and where $\chi_A(\cdot)$ is the indicator function of set A. Each cubic polynomial is usually expressed in the canonical form where it depends on four parameters a_i, b_i, c_i, d_i:

$$p_i(x) = a_i (x - x_i)^3 + b_i (x - x_i)^2 + c_i (x - x_i) + d_i. \tag{4.2}$$

Beside the passage through each node ($s(x_i) = y_i$), cubic splines are also required to have continuous first and second derivatives. Natural splines impose the additional smoothness constraint that the second derivative is null at the endpoints ($s''(x_1) = s''(x_N) = 0$).

A convolutional neural network is used to find the nodes of the splines (one for each color channel) that are the most suitable to enhance the input image. More precisely, we take the x coordinates to equally divide the $[0, 1]$ range (therefore $x_i = (i-1)/(N-1)$), while the y coordinates are computed by the network.

The architecture of the CNN is inspired by the work of [15], since it provided good performances on a similar problem. More popular and larger architectures would

Table 4.2 Structure of the convolutional neural network used to estimate the coordinates of the nodes of the splines. N denotes the number of nodes of the spline and $H \times W$ the size of the input image.

Stage	Operation	Output size
Pre-processing	Input	$H \times W \times 3$
	Bilinear resizing	$256 \times 256 \times 3$
Conv. Network	Conv. + ReLU	$63 \times 63 \times 8$
	Conv. + ReLU + B.N.	$31 \times 31 \times 16$
	Conv. + ReLU + B.N.	$15 \times 15 \times 32$
	Conv. + ReLU + B.N.	$7 \times 7 \times 64$
	Avg. Pooling	$1 \times 1 \times 64$
	Linear + ReLU	64
	Linear	$3N$
	Reshape	$N \times 3$
Post-processing	Identity Sum	$N \times 3$

probably be unsuitable for this task, due to the limited amount of training data. The downsampled version of the input image having size 256×256, is firstly processed by a set of convolutional layers followed by ReLUs. With the exception of the first block, all the activations are normalized through a Batch Normalization layer with a momentum of 0.01. After four of these blocks, we apply an average pooling with size 7×7 to reduce the feature map to a single feature vector. The last part of the architecture includes two fully connected layers separated by a ReLU that projects the activations into the splines node space. As a final step, we add the node positions $x_i = (i-1)/(N-1)$ to the output of the last fully connected layer. This last operation is inspired by the success of residual networks [51] and has the effect, when the parameters of the network are close to zero, to produce splines close to the identity function. This allows for a quicker training of the network and for an easier initialization of its parameters.

The output is a vector containing the N nodes of the 3 splines, therefore it's composed by $N \times 3$ elements. The architecture is summarized in Table 4.2.

4.3.2 Spline interpolation

For each color channel, we need to find the spline which interpolates the nodes estimated by the CNN. In other words we have to compute the coefficients a_i, b_i, c_i and d_i of each polynomial. Since there are $N-1$ polynomials we need $4N-4$ constraints to

determine all the coefficients. The interpolation constraints require that the polynomials pass through the nodes at their endpoints:

$$\begin{aligned} p_i(x_i) &= y_i, \\ p_i(x_{i+1}) &= y_{i+1}, \quad i \in \{1, \dots, N-1\}, \end{aligned} \tag{4.3}$$

and the continuity of the derivatives imposes that:

$$\begin{aligned} p_i'(x_{i+1}) &= p_{i+1}'(x_{i+1}), \\ p_i''(x_{i+1}) &= p_{i+1}''(x_{i+1}), \quad i \in \{1, \dots, N-2\}. \end{aligned} \tag{4.4}$$

Finally, natural splines require a null curvature at the extremities:

$$p_1''(x_1) = p_{N-1}''(X_N) = 0, \tag{4.5}$$

for a total of $4N-4$ constraints.

Following Bartels *et al.* we may express the constraints as a system of linear equations [12]. To this aim, it is useful to consider the expressions of the derivatives of the polynomials:

$$\begin{aligned} p_i'(x) &= 3a_i (x - x_i)^2 + 2b_i (x - x_i) + c_i, \\ p_i''(x) &= 6a_i (x - x_i) + 2b_i. \end{aligned} \tag{4.6}$$

By substituting equations (4.2) and (4.6) in the constraints (4.3) and (4.4) we obtain

$$\begin{aligned} a_i &= \frac{m_{i+1} - m_i}{6h}, \\ b_i &= \frac{m_i}{2}, \\ c_i &= \frac{y_{i+1} - y_i}{h} - \left(\frac{m_{i+1} + 2m_i}{6}\right) h, \\ d_i &= y_i, \end{aligned} \tag{4.7}$$

where $m_i = s''(x_i)$ is the second derivative of the spline at the nodes, and where $h = x_{i+1} - x_i = 1/(N-1)$ is the distance between two consecutive nodes (the expressions would be significantly more complex if the nodes were not equally spaced). Beside $m_1 = m_N = 0$ that is an obvious consequence of (4.5), the other values $\vec{m} = (m_2, m_3, \dots, m_{N-1})^T$ can be obtained by solving the following system:

$$A\vec{m} = \frac{6}{h^2} V\vec{y}, \tag{4.8}$$

defined in terms of the vector $\vec{y} = (y_2, y_3, \ldots, y_{N-1})^T$ and of the tridiagonal matrices

$$A = \begin{bmatrix} 4 & 1 & 0 & \cdots & 0 \\ 1 & 4 & 1 & \cdots & 0 \\ 0 & 1 & 4 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 4 \end{bmatrix}, V = \begin{bmatrix} -2 & 1 & 0 & \cdots & 0 \\ 1 & -2 & 1 & \cdots & 0 \\ 0 & 1 & -2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & -2 \end{bmatrix}. \tag{4.9}$$

After the computation of the matrix

$$B = \frac{6}{h^2} A^{-1} V, \tag{4.10}$$

the solution for $\vec{m}$ is given by the linear expression

$$\vec{m} = B\vec{y}. \tag{4.11}$$

Note that under the assumption that nodes are equispaced between 0 and 1 the matrix B depends only on the number of nodes N. Therefore, it can be computed in advance once this value has been set.

4.3.3 Color transformation

Once the nodes have been estimated upon a downsampled version of the input image and the three splines have been interpolated, we apply the color transformation to each pixel of the high resolution image. Given a value $\hat{x} \in [0, 1]$ for one of the three color components the transformed value is computed by applying the spline $s(\hat{x})$ for the corresponding color channel. This is done in two steps: first the index $\hat{\imath}$ of the polynomial is computed as

$$\hat{\imath} = \min\left\{1 + \lfloor \hat{x} \times (N-1) \rfloor, N-1\right\}, \tag{4.12}$$

then the value $\hat{y}$ is obtained by applying the selected polynomial

$$\hat{y} = p_i(\hat{x}). \tag{4.13}$$

The whole image enhancement procedure is summarized by Algorithm 2. Note that all operations are differentiable, and that they can be easily parallelized. Therefore the algorithm is suitable for an efficient implementation capable of running on the modern GPUs commonly used for deep learning applications.

Algorithm 2: The SpliNet image enhancement procedure.

Input: $H \times W$ color image I
Input: Number of nodes N
Output: $H \times W$ color image O
begin
  Compute B by following Equation (4.10);
  $\tilde{I} \leftarrow$ Donwsample(I);
  $(\vec{y}^{(r)}, \vec{y}^{(g)}, \vec{y}^{(b)}) \leftarrow \mathrm{CNN}(\tilde{I})$; `// Forward pass of the network`
  for $c \in \{r, g, b\}$ **do**
    $\vec{m} \leftarrow B\vec{y}^{(c)}$; `// Eq. (4.11)`
    for $i = 1$ **to** $N - 1$ **do**
      Compute a_i, b_i, c_i, d_i according to (4.7)
    end
    for $k = 1$ **to** H **do**
      for $l = 1$ **to** W **do**
        $\hat{\imath} \leftarrow 1 + \left\lfloor I_{kl}^{(c)} \times (N-1) \right\rfloor$; `// Eq. (4.12)`
        if $\hat{\imath} > N - 1$ **then**
          $\hat{\imath} \leftarrow N - 1$
        end
        $O_{kl}^{(c)} \leftarrow p_{\hat{\imath}}(I_{kl}^{(c)})$; `// Eq. (4.2)`
      end
    end
  end
end

4.4 Experimental setup

Measuring the performance of image enhancement methods is a challenging task. Beside the subjective human judgment, objective assessment procedures are limited by the amount of reference data available. The most common approach consists in measuring the difference between the images produced by the algorithms and the images retouched by human operators (hopefully chosen among experts in the field). The evaluation procedures reported in the literature differ in the sets of images used for training and test, in the performance metrics etc. Since there is not a commonly recognized standard for the evaluation of enhancement methods, we devised our own setup that is described in detail by the following sections. The dataset used in this work is the FiveK datset (see appendix A.4 for further datails).

Fig. 4.2 A sample from the FiveK dataset. For each raw image the dataset includes five different versions retouched by five expert photographers.

4.4.1 Metrics

Following [43], the first performance criterion considered is based on the difference in color appearance as perceived by a human observer. To this end, the ground truth and output images are converted in the perceptually uniform color space *CIELab*. Close attention should be paid to the conversion step of images into *CIELab* color space [41], since using the wrong transformations could easily bias the error calculation. In this work, since both the experts' ground truth images and the enhanced output images are in the *sRGB* color space, the standard equations from *sRGB* to *CIELab* as defined by International Electrotechnical Commission (IEC) [57] are used (see Appendix B).

The first color difference metric used is the $\Delta E_{76}(CIE76)$ that is calculated as l_2 distance in *CIELab* color space:

$$\Delta E_{76} = \sqrt{(L_1 - L_2)^2 + (a_1 - a_2)^2 + (b_1 - b_2)^2} = \sqrt{\Delta L^2 + \Delta a^2 + \Delta b^2}, \tag{4.14}$$

The second color difference metric used is the $\Delta E_{94}(CIE94)$ [29], that is a refinement of the first one and is computed as weighted l_2 distance in *CIELab* color space:

$$\Delta E_{94} = \sqrt{\left(\frac{\Delta L}{K_L S_L}\right)^2 + \left(\frac{\Delta C}{K_C S_C}\right)^2 + \left(\frac{\Delta H}{K_H S_H}\right)^2}, \tag{4.15}$$

where $\Delta C = C_1 - C_2 = \sqrt{a_1^2 - b_1^2} - \sqrt{a_2^2 - b_2^2}$, $\Delta H = \sqrt{\Delta a^2 + \Delta b^2 - \Delta C^2}$, $K_L = K_C = K_H = 1$, $S_L = 1$, $S_C = 1 + K_1 C_1$, $S_H = 1 + K_2 C_1$ with $K_1 = 0.045$ and $K_2 = 0.015$.

The third error metric used is the ΔL, that only considers the difference in luminance, thus discarding the color information. This is used to better understand how the luminance and color components contribute to the final ΔE error.

The last metric considered is the structural similarity index (SSIM) [116], that is a perception-based model that considers image degradation as perceived change in structural information. The SSIM is calculated on 11×11 sliding windows to produce a local distortion map, which is then averaged to obtain the final global SSIM index. Given the two sliding windows x and y respectively belonging to the luminance components of the input and ground truth images, the local SSIM index is computed as follows:

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{\left(\mu_x^2 + \mu_y^2 + c_1\right)\left(\sigma_x^2 + \sigma_y^2 + c_2\right)}, \tag{4.16}$$

where μ_x and μ_y are respectively the average of x and y, σ_x^2 and σ_x^2 respectively the variance of x and y, σ_{xy} the covariance of x and y; $c_1 = (k_1 L)^2$,$c_2 = (k_2 L)^2$ two variables to stabilize the division; L the dynamic range, $k_1 = 0.01$ and $k_2 = 0.03$.

4.4.2 Implementation and training

SpliNet has been implemented in the Python programming language using the Pytorch deep learning framework [90]. All the operations, including image preprocessing, downsampling, color space conversions, node estimation and spline interpolation have been integrated into an end-to-end deep learning module. Except where differently specified, we set the neural network to estimate ten nodes per spline.

The training procedure consisted of 40 000 iterations of the Adam learning algorithm [67] set to minimize the average ΔE_{76} between the output and the ground truth images. The learning rate was set to 10^{-4}, the weight decay was 0.1 and each mini-batch consisted of 20 images. The whole training procedure requires about 220 minutes of computation on a NVIDIA Titan Xp GPU.

The source code, the trained models and the data are publicly available in the github repository https://github.com/dros1986/neural_spline_enhancement.

Table 4.3 Accuracy in reproducing the test images retouched by expert C (for ΔE_{76}, ΔE_{94} and ΔL the lower the better, for SSIM the higher the better).

Method	ΔE_{76}	ΔE_{94}	ΔL	SSIM
Optimal gamma correction	17.05	12.31	9.14	0.793
Exposure [54]	16.98	13.42	9.54	0.872
CycleGAN [127]	16.23	12.30	9.20	0.835
Unfiltering [15]	13.17	10.42	6.76	0.922
HDRNet [43]	12.14	9.63	6.27	0.930
Pix2pix [59]	11.13	8.53	5.55	0.915
SpliNet	**10.70**	**8.17**	**5.52**	**0.942**

4.5 Single-user modeling

In the first experiment assessed how well the proposed method is able to learn to reproduce the retouching style of a single photographer. As commonly done in the state of the art we considered the images from expert C in the FiveK dataset.

In Table 4.3 we report the results obtained by our method measured in terms of ΔE_{76}, ΔE_{94}, ΔL, and SSIM. The table also reports the results obtained by recent methods in the state of the art retrained on our version of the training set by using the source code provided by their authors. As a further reference, we also add the errors obtained by a gamma correction where the gamma is chosen for each image with a grid search that minimizes the ΔE_{76}.

The first thing that can be noticed from the results reported in Table 4.3 is that the worst results are obtained by the optimal gamma correction; this means that learning just a gamma correction, even if optimal, is not enough to learn to reproduce the retouching style of a photographer; on the other side, this also means that all the considered methods learn a transformation that is more complex than a simple gamma correction. From the comparison with the state of the art it can be seen that the proposed method obtains the best results in terms of all the error metrics considered. The second best method, i.e. Pix2Pix [59], is very close in terms of ΔL luminance error but makes larger ΔE_{76} and ΔE_{94} colorimetric errors. Moreover, thanks to the smoothness and the globality of the spline-based transformation used, the proposed method has the largest SSIM index. This means that our method is the one that introduces the less amount of structural artifacts, which is the main drawback of GAN-based methods such as CycleGAN [127] and Pix2Pix [59]. The second best method in terms of SSIM index is HDRNet [43], that is the third one in terms of colorimetric error.

In Figure 4.3 we report some sample images processed by our method and by the state-of-the-art methods considered; for all of them the original raw image and the ground truth image retouched by expert C are also reported. It is possible to notice how Exposure [54] and CycleGAN [127] tend to produce visual results that are very different from the ground truth, while our method, HDRNet [43] and Pix2pix [59] are much closer. This shows how difficult is to learn from unpaired examples.

In order to show what is the typical shape of the splines learned by our method, some further examples of images processed by our method together with the corresponding learned splines are reported in Figure 4.4. It is possible to notice how the transformation is different for each image, and how for some images the splines are almost the same for each color channel, while for others are different.

4.5.1 Color distribution

In Table 4.5 we report the average histograms in *CIELab* color space for the images processed with our method and the ground truth images retouched by expert C. Three different histograms are reported, one for each color channel, all computed with bin size equal to five. To better show the differences between the two histograms they are reported using a logarithmic scale. From the plots it is possible to notice how our method does a pretty good job in predicting the luminance channel L, while it tends to produce images with a and b distributions more dense towards zero, resulting more conservative in terms of color saturation.

4.5.2 Dependence on the image content

The best enhancement for an image may depends on its semantic content. To identify the type of images that favor our method and those where the worst errors occur, we computed the performance metrics on homogeneous subsets of the test set. To do so we used the categorization provided with the FiveK dataset which divides images by subject, illumination, time of the day and location. The results in terms of average ΔE_{76} are reported in Figure 4.6. Low errors are obtained on outdoor images taken by day under a natural illumination. This kind of images usually do not present severe distortions such as strong color casts, exaggerated contrasts, over- or under-exposition, etc. Colors tend to be natural, and their distribution does not require a sophisticated adjustment. Moreover, these images are quite common so that it is relatively easy for the model to learn to reproduce a pleasant dynamic range using easily identifiable elements, such as the sky, as a reference.

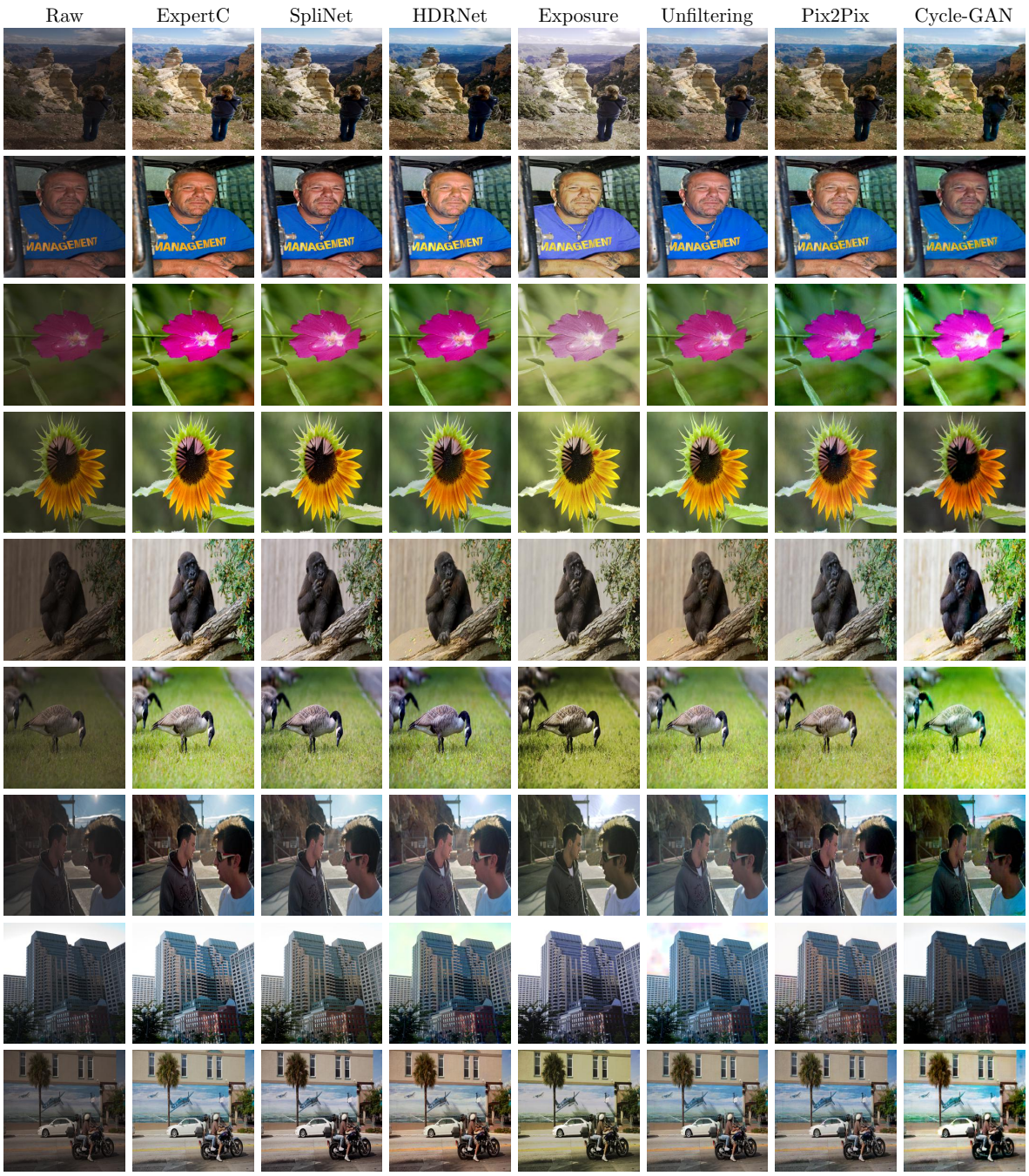


Fig. 4.3 Examples of test images enhanced by SpliNet and by other methods in the state of the art. The first column contains the input raw images, the second shows the target images retouched by expert C and the remaining columns report the results of the enhancement methods considered.

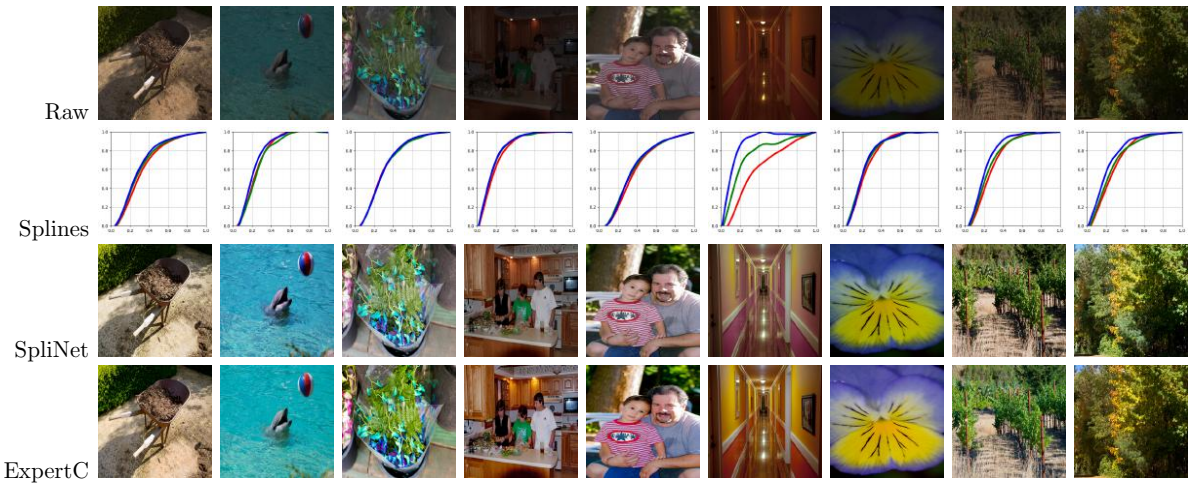


Fig. 4.4 Examples of images processed by SpliNet. From top to bottom the rows contain: input raw images, inferred splines color curves, SpliNet outputs and the ground-truth images retouched by expert C.

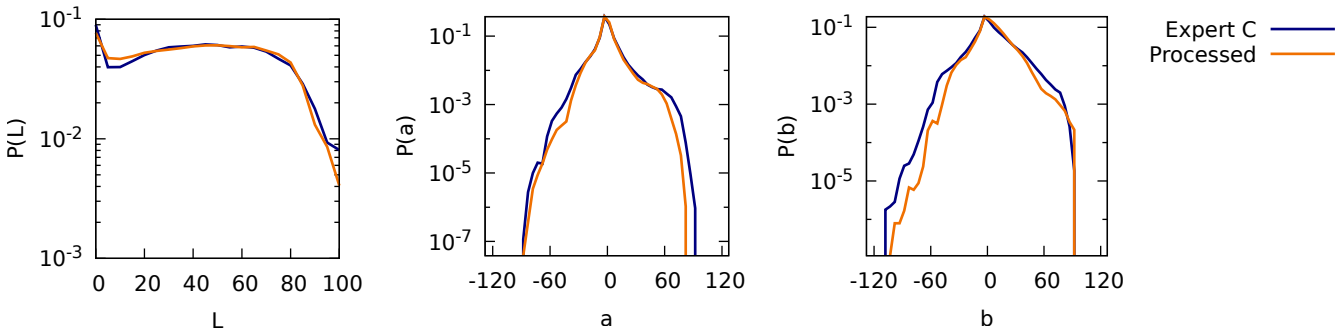


Fig. 4.5 Comparison of the distributions of the Lab color channels for the whole test set retouched by expert C and processed by the proposed method.

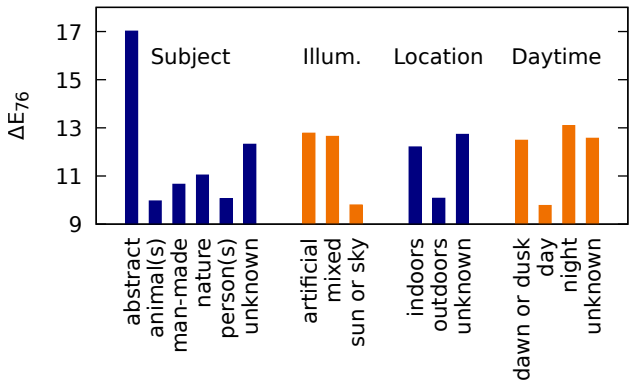


Fig. 4.6 Detail of the accuracy (average ΔE_{76}) in reproducing the test images retouched by expert C, measured on different subsets of the test set. Images are divided by subject, type of illumination, location and time of the day.

The most difficult images to enhance are those with an unusual content (e.g. abstract subject) or imaging conditions (e.g. taken by night). Beside being inherently more difficult to process, these kind of images have also few occurrences in the training set, making them even harder to enhance for a learning-based method.

4.5.3 Sensitivity analysis

We performed a sensitivity analysis of our method with respect to the number of spline nodes to be estimated. We plot in Figure 4.7 the ΔE_{76}, ΔE_{94}, and ΔL error values obtained by our method varying the number of spline nodes in the range $[5, 20]$ by steps of five. From the plot it can be seen that using a low number of spline nodes results in the worst performance, and then the performance starts to flatten when using ten nodes. Therefore in the experiments we used ten spline nodes, since it represents the best trade-off between accuracy and model complexity.

4.5.4 Processing time

One of the advantages of our method is that it allows to quickly process high-resolution images. The more complex operations are carried out on a low-resolution thumbnail and the final application of the splines scales linearly with the number of pixels in the input image. Figure 4.8 reports the actual processing time as a function of the size of the input image obtained by executing our method on a computer equipped with a NVIDIA Titan Xp GPU. The plot shows how for small images the processing time grows slowly. In this case, in fact, the most expensive operation is represented by the forward step of the convolutional neural network, which is executed on thumbnails of

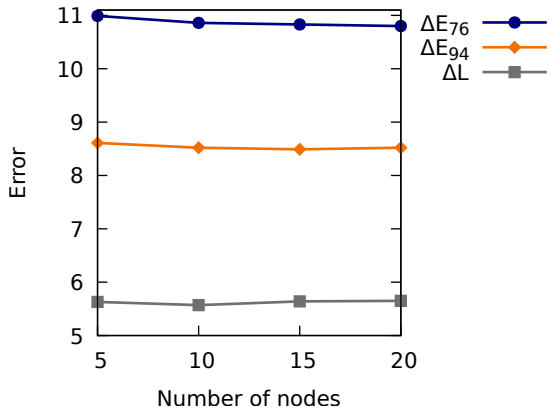


Fig. 4.7 Errors in reproducing the images retouched by expert C varying the number of nodes defining the splines.

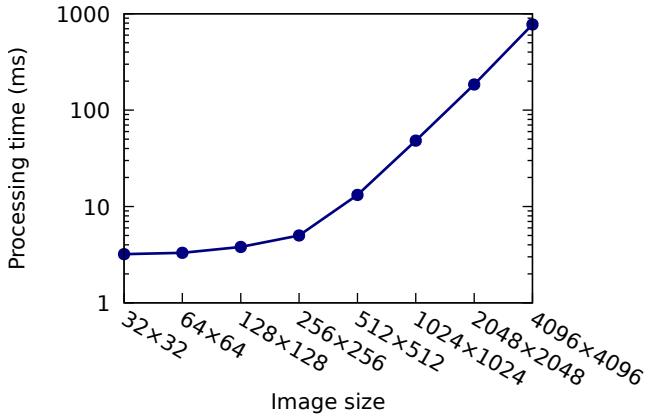


Fig. 4.8 Processing time (in milliseconds) taken to enhance images of various sizes. Each data point is the average over 100 runs.

256×256 pixels. For larger images, the total time is dominated by the application of splines. Even large images can be processed in reasonable times. For instance, it takes about 185 milliseconds to enhance a 2048×2048 image.

4.6 Multi-user modeling

For a completely automatic system, being able to accurately reproduce the retouching abilities of a professional photographer is certainly a remarkable result. However, in practice it doesn't matter how accurate the reproduction and how good the photographer are, such a system would leave many users unsatisfied. In fact retouching is not only a technical issue, but it is also a form of art, where subjective aesthetic judgments can be of cardinal importance. A very colorful style could be perceived as stunning by some users

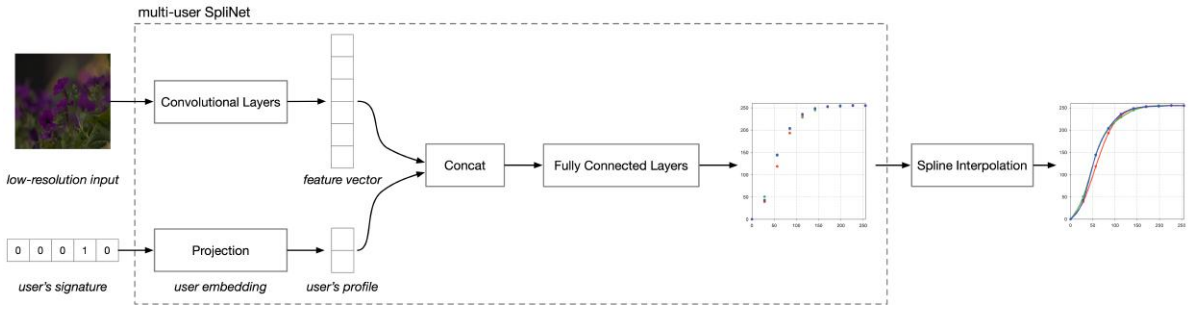


Fig. 4.9 Multi-user version of SpliNet. The input image is processed by taking into account the identity of the user, provided as a signature, and his preferences, encoded in a numerical profile.

and as exaggerated by others; similarly, a conservative style could be judged by different viewers as realistic and natural looking, or as too bland and dull. An ideal system for automatic image enhancement should be able to adapt its style to the taste of the user.

In order to make our system adaptive, we modified the model described in Section 4.3 to support the reproduction of the styles of multiple retouchers. In addition to the image to process, the neural network now takes as input a signature that uniquely represents a user. The signature is transormed in a profile of that user, consisting of a tuple of coordinates in a suitable user space. The profile is then injected in the network that will use it to adjust its output accordingly. More in details, a linear projection U maps the signature vector $\vec{s}$ into the user profile $\vec{p} = U\vec{s}$ which is then concatenated to the feature vector produced by the convolutional layers of the network, after the spatial average and before the fully-connected layers. The coefficients of the projection matrix U are learned together with the other parameters of the network.

The training process is performed by feeding to the network pairs of images and signatures and by requiring (via the loss function) that the output images match those retouched by the users identified by the signatures. By optimizing U, the training process places users in suitable positions in the user space. For instance, users with a similar retouching style are expected to be located near each other (i.e. they will have similar profiles). A low-dimensional user space forces the network to model the style of the users by identifying their common preferences and by highlighting their main stylistic traits. A scheme of the multi-user version of the system is reported in Figure 4.9.

We trained the modified model on the training set of the FiveK dataset by using the 4000 training images retouched by all the five available experts. We used five-dimensional one-hot vectors as signatures (the signature of expert A was $\vec{s}_A = \langle 1, 0, 0, 0, 0 \rangle$, and so on), and we set the user space to be two-dimensional (to increase its dimensionality we

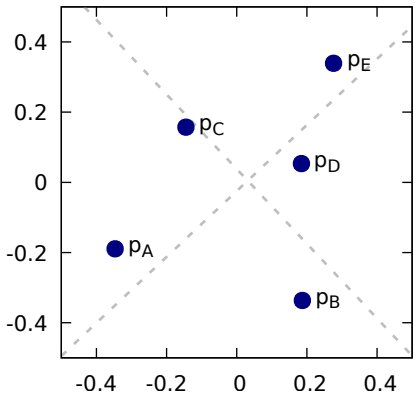


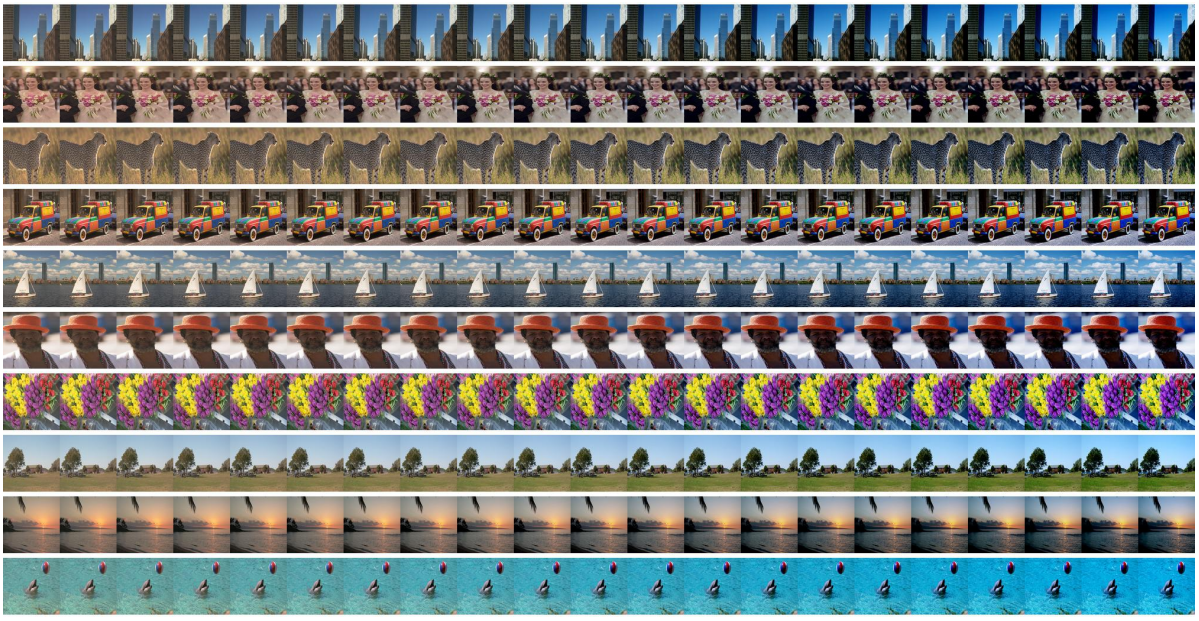
Fig. 4.10 Graphical representation of the user space. The points represent the profiles of the five experts, while the dashed lines represent the two principal components (the first goes approximately int the p_A–p_E direction, while the second is in the p_B– p_C direction).

would probably need more experts). The number of training iterations was five times that used to train the single expert version.

As a result of the learning process we obtained, together with the trained model, a definition of the user space. Figure 4.10 shows the positions in that space of the profiles of the five experts. To better understand the user space we performed a principal component analysis of the five profiles and we observed how moving along the two principal components changes the output of the network. Figure 4.11 reports the outcome of this experiment on a small sample of test images. It is evident that by moving in the direction of the first principal component it is possible to adjust the brightness of the image. For the second component, instead, we observe images with warmer tones on one side and a prevalence of cold colors on the other. Therefore we can say that the second component captures the chromatic element of the retouching style. By combining these directions (i.e. by moving in the user space) it is possible to obtain a variety of retouching styles including, but not limited to, those of the five experts used during the training.

We evaluated the learned model on the 1000 test images repeating the evaluation five times, each time selecting the profile of a different expert. Table 4.4 reports the results obtained. By comparing the performance of the single- and the multi-user models we notice how this latter does not incur in any loss in performance. On the contrary, the multi-user version slightly outperforms the single-user version. This is not too surprising, if we consider that, like in multitask learning [26], the multi-user model is encouraged to learn more robust intermediate features in order to be able to reproduce different styles.

(a)



(b)

Fig. 4.11 Examples of image enhancement performed by the multi-user model by moving the user profile along the first (a) and the second (b) principal components in the user space. For each principal component the pictures span a range of ± 10 standard deviations from the center of the user space.

Table 4.4 Performance of the single- and multi-user models.

Model	User	ΔE_{76}	ΔE_{94}	ΔL
Single-user	Expert A	11.32	9.48	7.40
	Expert B	9.01	7.27	4.87
	Expert C	10.70	8.17	5.52
	Expert D	10.93	9.16	7.03
	Expert E	10.78	8.44	5.73
Multi-user	Expert A	10.91	9.09	7.16
	Expert B	8.69	7.01	4.88
	Expert C	10.50	7.97	5.61
	Expert D	10.63	8.93	6.97
	Expert E	10.33	8.03	5.52

From the computational point of view the difference between the multi- and the single-user versions is negligible: the multi-user model includes just 138 additional parameters: 10 in the projection matrix U and 128 extra coefficients in the first fully-connected layer.

4.6.1 Modeling a new user

Given a new user it would be desirable to be able to adapt the system to his style without asking him to provide a whole new training set of retouched photographs. Instead, it would be very convenient to reproduce his retouching style just by choosing a suitable profile for him in the user space. To this end, we devised two strategies: the *voting strategy* and the *fitting strategy*.

In the voting strategy multiple retouched versions of the same image are shown to the new user and he is asked to choose the one he likes most. This is repeated multiple times with different images keeping track of the answers. Let K be the number of users on which the multi-user model has been trained and let $\vec{p}_i$ be their profiles in the user space $i = 1, \ldots, K$. The profile $\vec{p}^{(V)}$ for the new user is computed from the number N_i of times in which he chose an image retouched by user i:

$$\vec{p}^{(V)} = \frac{\sum_{i=1}^{K} \vec{p}_i N_i}{\sum_{i=1}^{K} N_i}. \tag{4.17}$$

In practice the new profile is a convex combination of the known ones, where the coefficients are the fractions of votes expressed by the new user.

In the fitting strategy the new user is given a set of m images $I_1, \ldots, I_m$ and is asked to retouch them according to his preferences, producing the images $\hat{I}_1, \ldots, \hat{I}_m$. Then his profile $\vec{p}^{(F)}$ is chosen by minimizing the average dissimilarity between the output of the network and the retouched images:

$$\vec{p}^{(F)} = \arg\min_{\vec{p} \in \mathbb{R}^2} \frac{1}{m} \sum_{i=1}^{m} d(\hat{I}_i, \text{SpliNet}(I_i, \vec{p})), \tag{4.18}$$

where $d(\cdot, \cdot)$ is a dissimilarity measure and $\text{SpliNet}(I_i, \vec{p})$ is the output of the method when fed with the input image I_i and when injected with the user profile $\vec{p}$. To find a solution to this optimization problem we initialize $\vec{p}$ with random values and we perform several iterations of the gradient descent algorithm by keeping all the parameters of the network fixed and by updating each time only the profile $\vec{p}$.

To assess the two modeling strategies we performed several simulations in which one of the experts in the FiveK dataset plays the role of the new user, and the other four act as the known users. To do so, we retrained the multi-user model five times, each one by excluding from the training set the images retouched by one of the experts. Then, in each simulation we selected a given number m of training images and applied the voting or the fitting strategy to obtain the user profile of the expert whose images were excluded from training. In the case of the voting strategy for each image the vote was assigned to the expert whose retouched version was less dissimilar from that retouched by the target expert. In the case of the fitting strategy we simply used in (4.18) the images retouched by the target expert. We tried with different number m of images and for each value of $m \in \{1, 3, 10, 30, 100, 300, 1000\}$ we carried out ten simulations for both the strategies. As a dissimilarity measure we used the average ΔE_{76}. For the fitting strategy we performed 100 iterations of gradient descent with a learning rate of 0.1.

The outcome of the simulations is summarized in Figure 4.12. With the voting strategy we obtained promising performance, never far from those obtained by the model trained on all the five experts (i.e. those in Table 4.4). Its behavior is very stable as the worst case is relatively close to the average one. The voting strategy presents one major weakness: the estimated user profile is restricted to be within the convex hull of the profiles of the known users. This means that when the optimal profile lies outside the convex hull we cannot obtain a good approximation. For instance, if we refer to Figure 4.10 this happens for expert A, since it cannot be expressed as a combination of the other four. In other words, the voting strategy cannot extrapolate the style of the new user outside the range of styles seen during training. This weakness is reflected in the

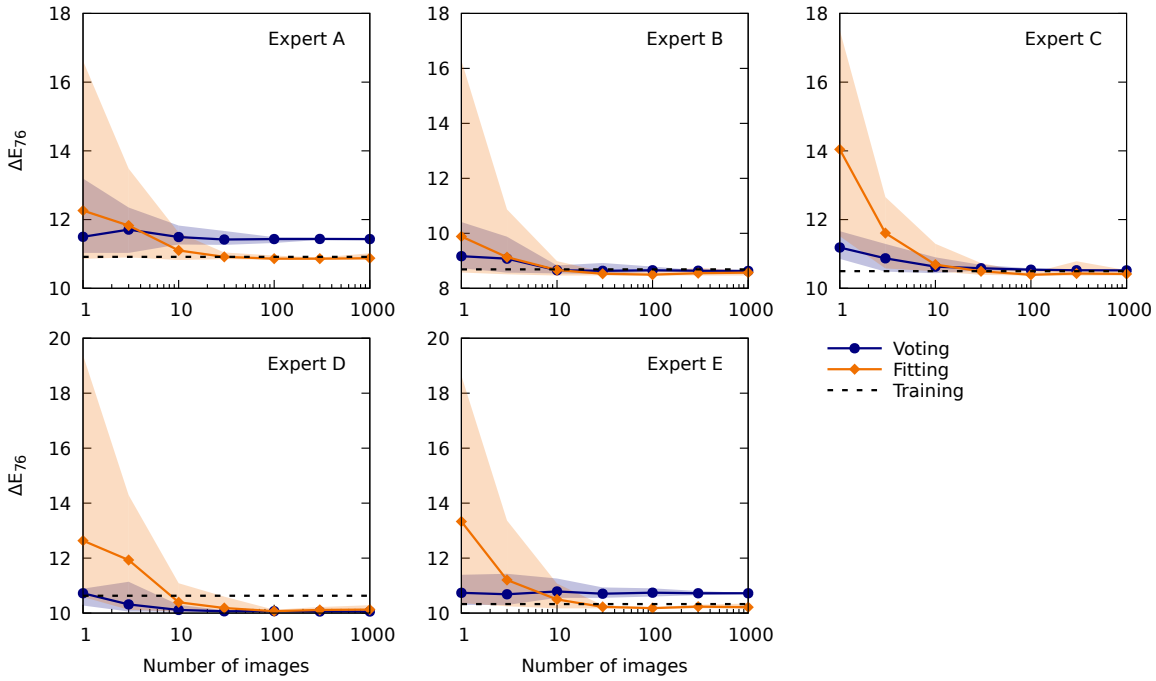


Fig. 4.12 Performance obtained on the test set by modeling each expert with the voting and the fitting strategies. Each plot reports the mean ΔE_{76} over ten simulations, varying the number of images used to model the user. The boundaries of the shaded areas represents the best and the worse simulations, while the dashed line reports the performance obtained for the target expert with the multi-user model trained on all the experts.

plots: only for experts B, C and D the voting strategy allows to match the performance of the fully trained model.

The fitting strategy, instead, poses no restrictions to the placement of the new profile in the user space. In fact, in all the cases it allowed to obtain performance that are equal or even slightly better than those obtained by the trained model, provided that at least ten images are used to fit the user profile. However, the lack of restrictions makes this strategy unreliable when a small number of images are used. In the worst case the fitting strategy can perform very poorly if less than ten images are used.

In terms of performance, the outcome of the simulations provided a clear recommendation: the voting strategy should be preferred when three or less images are used, otherwise the choice should fall on the fitting strategy. Beside that, the voting strategy has a couple of advantages over the fitting strategy: it is computationally less intensive, and it does not require the new user to actually retouch the images since he is just asked to choose among those already retouched by other users.

Part IV

Anomaly Detection

Chapter 5

Introduction To Anomaly Detection

"Science is the systematic classification of experience."

George Henry Lewes

The state of the art for anomaly detection is composed by several methods, each one of them searching anomalies in different target products. The design of the methods depends of course from the structure and the appearance of the types of products under inspection, however, some of those methods can be used on different domains with similar properties without problems. Figure 5.1 shows a possible topology for cathegorizing anomaly-detection methods upon the assumpion they make on the normal signal (here also called *background*). If the background is *homogeneous* (see section 1.2 for reference) and it's *frequency-separable*, a solution based on the analysis in the frequency domain through the Fourier transform is preferred. Tsai and Hsieh [112] for example, assume that anomalies are separated in the frequency domain with respect to the background. Therefore they propose a method based on background removal obtained through a frequency cutoff. Similarly, Perng et al. [91] propose a method for internal thread anomaly detection. An internal thread is a spiral screw pattern cut into the inner surface of a hollow cylinder. The inside of the threads can be considered as a repetitive pattern, therefore it can be easily removed by deactivating the corrispondent harmonics in the frequency domain. The remaining part of the image will be the defects (if any) present in the inside of the thread. Tout et al. [111] relax the constraint of having the normal signal at a specific frequency by finding specific spots of the wheels under analysis in the frequency domain and fitting the model (rotation and scaling) to the input wheel image. In contrast to previous frequency-based methods, Aiger and Talbot [2] do not assume that the normal signal has a specific frequency. They just assume that the background is totally regular and therefore can be detected through the Phase Only Transform (PHOT)

which correspond to the Discrete Fourier Transform (DFT), normalized by the magnitude. The PHOT removes any regularities (the background), leaving only the anomalies.

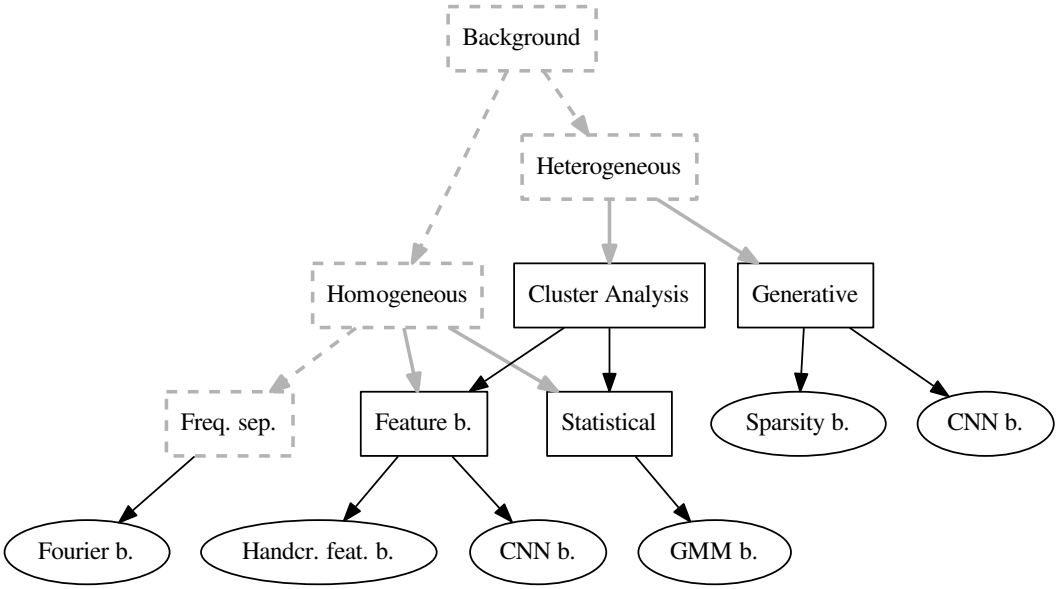


Fig. 5.1 Topology of anomaly-detection methods. In dashed boxes the hypotesis made on the normal class (here referred as *background*), in boxes the abstract families of algorithms and finally, in baloons the real families of algorithms. Dashed arrows indicate extensions of the previous concept, full gray arrow the trailing family of algorithms.

If the background is homogeneous but it's not *frequency-separable*, two types of strategies are adopted in the state of the art, the *feature-based* and the *statistical-based* methods. Among the *feature-based* methods there is the one proposed by Honda and Nayar [53]. They find anomalies in arbitrary images, with the assumption that images under inspection have regular patterns. In this context, an anomaly is defined as a disruption of these patterns. The basic idea is to compute a probability density for sub-regions in an image, conditioned upon the areas surrounding the sub-regions. Pseudo non-parametric correlation is used to group sets of similar surrounding patterns, from which a probability for the occurrence of a given sub-region is derived. This method cannot be applied as-is in a industrial quality inspection system, because it makes the assumption that the good (normal) signal is always present in the image under inspection and that it covers the majority of the image. This assumption cannot be held in a production chain, where can occur huge disasters that totally affect the products.

Margolin et al. [81], adopt a similar approach for saliency in the context of object detection. They investigate a system based on Principal Component Analysis (PCA) for emerging the distinctive patches in an image. In a context of anomaly detection this system can be used to make emerge small portions of the signal that are not compliant with the rest of the image. Here again there is the same problem of Honda and Nayar [53]: this system in fact, is based on the assumption that the normal signal covers the majority of the image. Furthermore, there is no knowledge of how the normal signal is done. In fact, this method does not make any type of learning or training of what is considered normal. Among the *statistical-based* algorithms, there is the one proposed by Du and Zhang [37] that uses a gaussian background model from 2×2 image patches in hyperspectral image. Once the background model $(\mu, \sum)$ is fitted, the anomalous patches are detected using a threshold on the Mahalanobis distance:

$$d(p_i) = \sqrt{(p_i - \mu)\sum^{-1}(p_i - \mu)} \tag{5.1}$$

Goldman and Cohen [45] propose a method for sea-mine detection. This method proposes an iterative schema to gradually reducing the false alarm rate while maintaining a high probability of detection. It performs a background estimation in a local feature space of principal components. Anomalous pixels are detected evaluating their Mahalanobis distance with the Gaussian. Methods such as [37] and [45] that directly use pixels values are limited and can be used only when the background composition is simple. Tarassenko et al. [110] perform mass identification in mammograms by modelling normal images with five types of features (standard deviation, texture correlation measure, edge gradient, ratio volume/area, contrast between contour and surrounding areas normalized to have equal weight) and assuming that abnormalities are uniformly distributed outside the boudaries of normality, defined using a gaussian mixture model. By clustering they define different areas of normality, each one with its own gaussian mixture model. Here again Mahalanobis distance is used as a distace metric. Xie and Mirmehdi [120] learn a texture model based on texton theory [61]. They assume that image patches follow a Gaussian model. They firstly cluster the various patches and then model each cluster through a density mixture model, characterized by the mean and the covariance of each cluster. In this context, each cluster model is called *textural examplars*. Grosjean and Moisan [47], instead of using directly the pixels values as features, it performs feature extraction through a convolution of white Gaussian noise with an arbitrary kernel (coloured noise) and then model the background as a Gaussian stationary process. Table 5.1 highlights the characteristics of the just decribed statistical approaches.

Table 5.1 Characterization of statistical approaches

Method	Feat.		Clust.	Pyr	Dist.	
	pixels	*handcrafted*	*cluster anal.*	*pyram.*	*Mahalanobis*	*other*
Du and Zhang [37]	✓				✓	
Goldman and Cohen [45]	✓			✓	✓	
Tarassenko et al. [110]		✓	✓		✓	
Xie and Mirmehdi [120]	✓		✓		✓	
Grosjean and Moisan [47]		✓			✓	

If the background is more complex and is *heterogeneous*, some researchers try also different techniques, such as the *generative approach*. This type of approach is very interesting and has the aim to create a generator that is able to create an anomaly-free version of the input image (operation called *restoration*) and later compare the result with the original image, that can contain anomalies. Basically these methods try to project an inquiry input image and the manifold of normal samples. Two different technologies are used in this context: the ones based on sparse coding and the ones based on neural networks. Boracchi et al. [20] exploit sparse coding to restore the input image. In their method, the background model is a learned dictionary of patches from an anomaly-free dataset. In test phase the inquiry image is reconstructed through a linear combination of the words of the dictionary. This dictionary is built through sparse coding. The degree of abnormality of an inquiry patch is determined by the reconstruction error and the $L1$ distance between the weighting coefficients used to weight the words in the linear sovraimposition with respect to the ones belonging to the train set. This method, however, cannot work with medium to big patches because of the reconstruction routine adopted. It becomes extremely hard to find base-patches that, combined together with a weighted sum, are able to represent the variation of a bigger patch. Hawkins et al. [50] instead, use a Replicator Neural Network (an fully-connected autoencoder with tanh activations between hidden layers) to regenerate the input image and getting rid of eventual anomalies. The degree of abnormality, here called *Outlier Factor*, is calculated as the average reconstruction error for each pixel, calculated through the L2-distance. An and Cho [4] use a variational autoencoder rather than a plain autoencoder. The main difference between using a straight autoencoder versus the use of a variational autoencoder is that in the first case, what the network learns is a compact representation of the input

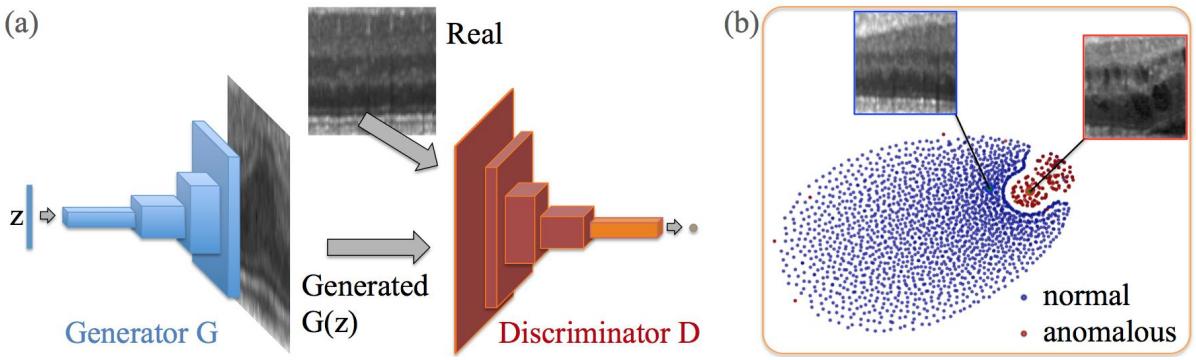


Fig. 5.2 (a) Deep convolutional generative adversarial network (DC-GAN) adopted for training the generator used in Schlegl et al. [99] for marker discovery. (b) t-Distributed Stochastic Neighbor Embedding (t-SNE) of the images in the feature space given by the last convolutional layer of the discriminator (highlighted in orange in subfigure (a)). Note that this space is easily separable since normal and anomalous samples (respectively depicted in blue and red) reside on different clusters.

instance, while in the second case the autoencoder learns the parameters of a probability distribution representing the data. To measure the degree of abnormality, instead of the reconstruction error, they use the reconstruction probability, which has a theoretical background (in constrast to plain autoencoder, which is just deterministic). Schlegl et al. [99] use a Generative Adversarial Network (specifically a DCGAN, see section 2.2 for more details) to create a CNN-based generator that, once trained, it learns the mapping $G(z) = z \rightarrow x$ from latent space representations z to realistic (normal) images x. Since the latent space has smooth transitions [93], the sampling from two points close in the latent space generates two visually similar images. They key idea of this method is, given a query image x, to find its representation z in the latent space that corresponds to an image $G(z)$ that is the visually similar to the query image and that is located in the manifold of normal images. Image 5.2 (b) shows this concept. An anomalous image is back-projected in the latent space and the closest latent representation of a normal sample is chosen as the most similar non-anomalous sample. This last sample in the form of image is then compared with the anomalous image to segment the non-compliant parts. Note that the back-projection of anomalous samples generates a cluster of latent representations that forms a separate cluster with respect to the ones corresponding to normal samples. In the figure, the latent space is showed in two dimensions through the use of t-Distributed Stochastic Neighbor Embedding (t-SNE), a technique created by Maaten and Hinton [80] for dimensionality reduction useful for the visualization of high-dimensional datasets. The problem is that GANs do not automatically yield the inverse mapping $\mu(x) = x \rightarrow z$. To solve this problem, they use an optimization process

that, starting from a random initialization of the latent representation z_0, it uses gradient descent to find the latent representation z that generates an image $G(z)$ that is the closest to the query image, but is anomaly-free. The loss function used to generate this image is composed by two losses, the *residual loss* R that ensures that the image is visually similar to the query image and the *discriminator loss* D, which ensures that $g(z)$ is in the manifold of normal images. The *residual loss* is an $L1$ loss. The anomaly score of the inquiry image will be a linear combination of these two losses at their last iteration in the optimization process, therefore:

$$A(x) = (1 - \lambda)\, R(x) + \lambda D(x) \tag{5.2}$$

Chapter 6

Feature-Based Methods For Anomaly Detection

"No good is ever done to society by the pictorial representation of its diseases."

John Ruskin

6.1 Introduction

Among all families of anomaly-detection algorithms, we present a new *feature-based* method based on cluster analysis that is able to adapt to all kinds of background (*homogeneous or heterogeneous*, see figure 6.1 for more details) and is able to work with just few images of normal samples (less then 10). This is because methods based on the assumption that the background is *heterogeneous* are able to work also on *homogeneous* backgrounds. In fact, the *homogeneous* ones, because of their simplicity, are straightforward to detect and can be seen as an *heterogeneous* background where there is only one pattern. In practice, for algorithms based on cluster analysis, this means that *homogeneous* backgrounds can be modeled with one single cluster. The presented method is based on deep learning and represents the state-of-the-art for the detection of *highly localized anomalies* in nanofibrous materials [86].

6.2 Anomaly detection and localization

Let us define an image $\mathbf{I}$ as a matrix of size $w \times h \times c$ of values $\in \mathbb{N}$. The variable h is the height, w is the width and c is the number of color channels which is equal to 3 in

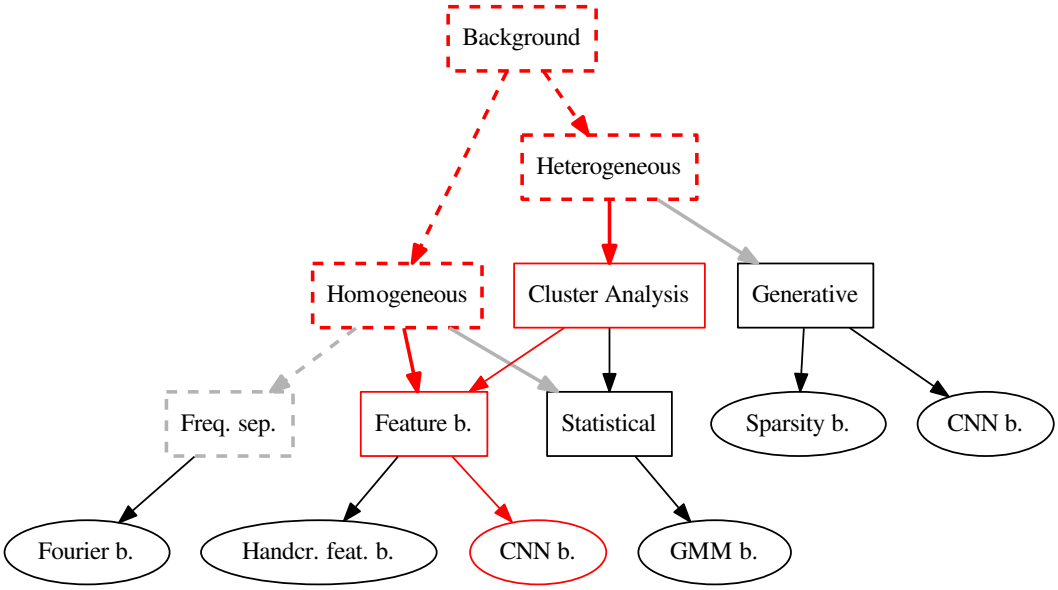


Fig. 6.1 The method presented in this chapter is able to work with homogeneous as well as heterogeneous backgrounds.

the case of Red, Green and Blue (RGB) images. Each element or pixel of the image $\mathbf{I}(p)$ at position p within the matrix of size $w \times h$, is a triplet of RGB values $\in \mathbb{N}$. Let us define $\Omega_{\mathbf{I}}$ as the binary mask of anomalies for the image $\mathbf{I}$. $\Omega_{\mathbf{I}}$ is a matrix of size $w \times h$ and each element at position p is such that:

$$\Omega_{\mathbf{I}}(p) = \begin{cases} 0 & \text{if } \mathbf{I}(p) \text{ is not an anomalous element of the image } \mathbf{I} \\ 1 & \text{if } \mathbf{I}(p) \text{ is an anomalous element of the image } \mathbf{I} \end{cases} \tag{6.1}$$

Given an image $\mathbf{I}$ and its mask of anomalies $\Omega_{\mathbf{I}}$, the anomaly detection problem is defined as the problem of automatically finding the binary mask $\widetilde{\Omega_{\mathbf{I}}}$ which best approximates the reference mask of anomalies $\Omega_{\mathbf{I}}$. The aim is thus twofold: a) to identify the largest number of anomaly pixels in $\mathbf{I}$; b) to cover all the anomalous regions in $\mathbf{I}$. An example of an image $\mathbf{I}$ containing anomalous elements, a binary mask of anomalies $\Omega_{\mathbf{I}}$, an approximated mask of anomalies $\widetilde{\Omega_{\mathbf{I}}}$ are represented in Fig. 6.3.

The basic assumption for an automatic method for anomaly detection is that the method is trained with normal (i.e., anomaly free, see Figs. 6.2(a) and (b)) images while it is tested with normal and anomalous images (see last Figs. 6.2(c) and (d)).

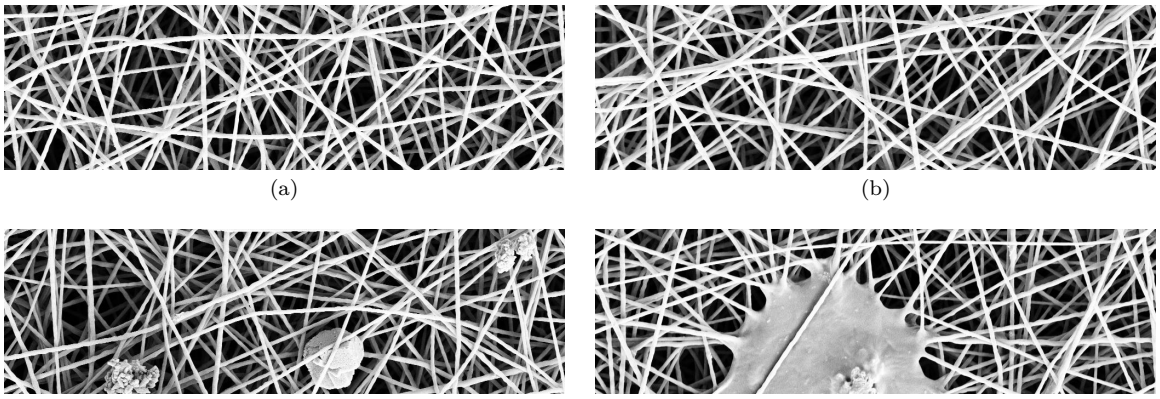


Fig. 6.2 SEM images of nanofibrous materials. (a) and (b) samples without anomalies. (c) and (d) samples containing fine- and coarse-grained anomalies.

Fig. 6.3 (a) The input image $\mathbf{I}$. (b) The binary mask of anomalies $\Omega_{\mathbf{I}}$. White pixels represent anomalies. (c) Estimated mask of anomalies $\widetilde{\Omega_{\mathbf{I}}}$. White pixels represent anomalies. (d) Difference between $\Omega_{\mathbf{I}}$ and $\widetilde{\Omega_{\mathbf{I}}}$ overlaid on the test image. Green pixels represent *true positives*, red pixels represent *false positives*, blue pixels represent *false negatives*, no color pixels represent *true negatives*.

Hereinafter, we denote the set of normal images used for training the method as $\mathcal{I}^{train}$, the set of normal images used for validating the method as $\mathcal{I}^{val}$, while we denote the set of anomalous images used for testing the method as $\mathcal{I}^{test}$. A method for anomaly detection learns a model of normality $\mathcal{M}(\theta)$ from the training images $\mathcal{I}^{train}$, where θ represents the free parameters of the model. The model $\mathcal{M}$ is inferred and used to assign an anomaly score $z(\mathbf{x})$ to previously unseen test data $\mathbf{x}$. Larger anomaly scores $z(\mathbf{x})$ correspond to a higher abnormality with respect to the model of normality. The final classification of the test pattern is obtained by defining a threshold th such that the test pattern $\mathbf{x}$ is classified "abnormal" if $z(\mathbf{x}) > th$, or "normal" otherwise. The equation $z(\mathbf{x}) = th$ defines a decision boundary for the anomaly detection method [92]. The threshold th is found by exploiting the set $\mathcal{I}^{val}$ that contains only images without anomalies.

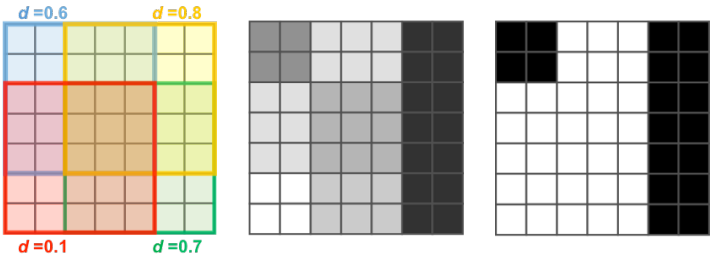


Fig. 6.4 Simulated example of the map $\Theta_{\mathbf{I}}$ and corresponding binary mask of anomalies $\widetilde{\Omega}_{\mathbf{I}}$. Here $w_t = h_t = 5$ and stride $s = 3$ and the variable d represents the value of the average visual similarity between the local patch and the most similar subregions of the dictionary $\mathcal{W}$.

6.3 Proposed method

The proposed anomaly detection method is region-based. The input image $\mathbf{I}$ is divided in a set of T subregions (or patches) of size $w_t \times h_t$ by following a regular grid sampling strategy with a stride s. The method takes each patch as input and it firstly computes its degree of abnormality and later it combines the response achieved on each subregion in order to have a map of anomalies $\Theta_{\mathbf{I}}$ that is then thresholded to obtain the binary map $\widetilde{\Omega}_{\mathbf{I}}$. The degree of abnormality of a patch is obtained by computing the visual similarity between the given patch and a reference dictionary $\mathcal{W}$ of normal subregions taken from normal images belonging to $\mathcal{I}^{train}$. The visual similarity is obtained by averaging the euclidean distances between the feature vector extracted from a given subregion and the feature vectors extracted from the m most visually similar subregions of the dictionary. Each subregion is selected with a stride s, so each pixel of the image $\mathbf{I}$ may belong to different partially overlapping subregions. In this case, the degree of abnormality of each pixel is obtained by averaging the degree of abnormality of each corresponding subregion. Finally, the mask of anomalies $\widetilde{\Omega}_{\mathbf{I}}$ is obtained by thresholding $\Theta_{\mathbf{I}}$ with th. Fig. 6.4 shows an example of the map $\Theta_{\mathbf{I}}$ obtained for $w_t = h_t = 5$ and stride $s = 3$ and the corresponding binary mask of anomalies $\widetilde{\Omega}_{\mathbf{I}}$.

6.3.1 Feature extraction

A huge variety of features have been proposed in literature for describing image visual content. They are often divided into hand-crafted features and learned features. Hand-crafted descriptors are features extracted using a manually predefined algorithm based on the expert knowledge. Learned descriptors are features extracted using Convolutional Neural Networks (CNNs) [31, 84].

CNNs are a class of learnable architectures used in many domains such as image recognition, image annotation, image retrieval etc. [100]. CNNs are usually composed of several layers of processing, each involving linear as well as non-linear operators, that are learned jointly, in an end-to-end manner, to solve a particular tasks. A typical CNN architecture for image classification contains several convolutional layers followed by one or more fully connected layers. The result of the last fully connected layer is the CNN output. The number of output nodes is equal to the number of image classes [68].

A CNN that has been trained for solving a given task can be also adapted to solve a different task. In practice, very few people train an entire CNN from scratch, because it is relatively rare to have a dataset of sufficient size. Instead, it is common to take a CNN that is pre-trained on a very large dataset (e.g. ImageNet, which contains 1.2 million images with 1000 categories [35]), and then use it either as an initialization or as a fixed feature extractor for the task of interest [94, 114]. In the latter case, given an input image, the pre-trained CNN performs all the multilayered operations and the corresponding feature vector is the output of one of the last network layers [114]. The use of CNNs as feature extraction method has demonstrated to be very effective in many pattern recognition applications [94, 85, 14, 32].

The first notably CNN architecture that has showed very good performance upon previous methods on the image classification task is the AlexNet by Krizhevsky et al. [68] After the success of AlexNet, many other deeper architectures have been proposed such as: VGGNet [104], GoogleNet [108] and Residual Networks (ResNet) [51]. ResNet architectures demonstrated to be very effective on the ILSVRC 2015 (ImageNet Large Scale Visual Recognition Challenge) validation set with a top 1- recognition accuracy of about 80% [51].

In this chapter I use CNN-based features obtained by exploiting a deep residual architecture. Residual Network architectures are based on the idea that each layer of the network learns residual functions with reference to the layer inputs instead of learning unreferenced functions. Such architectures demonstrated to be easier to optimize and to gain accuracy by considerably increasing the depth [51]. For instance, the ResNet-50 architecture is about 20 times and 8 times deeper than AlexNet and VGGNet respectively.

In particular, the network architecture adopted in this chapter is based on the ResNet-18 architecture which represents a good trade-off between depth (that is computational time) and performance. The network architecture includes 5 convolutional stages (see Tab. 6.1 for further details). The network is pre-trained on the set of images defined by the ILSVRC 2015 challenge. The goal of this challenge is to identify the scene and object categories depicted in a photograph. The total number of categories is 1000.

Table 6.1 ResNet-18 Architecture

layer name	output size	ResNet-18
`conv1`	$112 \times 112 \times 64$	7×7, 64, stride 2
`conv2_x`	$56 \times 56 \times 64$	3×3 max pool, stride 2
		$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$
`conv3_x`	$28 \times 28 \times 128$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$
`conv4_x`	$14 \times 14 \times 256$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$
`conv5_x`	$7 \times 7 \times 512$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$
`average pool`	$1 \times 1 \times 512$	7×7 average pool
`fully connected`	1000	512×1000 fully connections
`softmax`	1000	

Although the network is pre-trained on scene and object images, it has demonstrated, in preliminary experiments, to work much better than a ResNet-18 pre-trained on texture images [33, 32]. The visual appearance of textures is certainly more similar to the visual appearance of the SEM images considered in this chapter. Notwithstanding this, the performance obtained by exploiting the texture-domain network are much worse than the performance obtained using a scene- and object-domain one. Actually, recognizing scenes and objects is more complicated than recognizing textures, and thus the network trained to recognize scenes and objects is more capable of recognizing unexpected anomalous patterns within SEM images.

Given the network, the output of a given layer is linearized to be used as feature vector. We experiment the use of two different layers of the network: the linearized output of the fifth convolutional stage (that is `conv5_x`) and the output of the average pooling layer (that is `avgpool`). The size of the feature vector is 25088 (that is $7 \times 7 \times 512$) in the case of the `conv5_x` layer and 512 in in the case of the `avgpool` layer. The size of the feature vector affects the computational cost, then the size is therefore reduced by applying dimensionality reduction techniques such as Principal Component Analysis.

6.3.2 Dictionary building

The degree of abnormality of a patch is obtained by computing the visual similarity between the given patch and a reference dictionary $\mathcal{W}$ of normal subregions. The dictionary is built from the set of training images $\mathcal{I}^{train} = \{\mathbf{I}_1, \cdots, \mathbf{I}_L\}$. For each image $\mathbf{I}_l$, T patches $\{\mathbf{P}_1, \cdots, \mathbf{P}_T\}$ of size $w_t \times h_t$ are extracted following a regular grid and using a stride s. The total amount of patches extracted from the whole training set $\mathcal{I}^{train}$

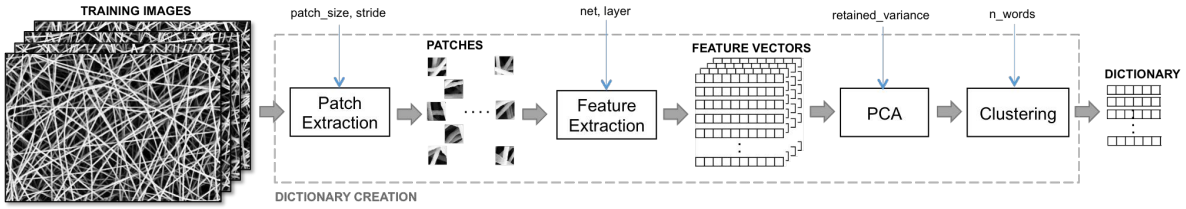


Fig. 6.5 Creation of the dictionary.

is $L_T = L \times T$. The feature extraction module computes for each patch $\mathbf{P}_t$ a vector of features of size N, $\mathbf{f}_t = \{f_1^t, \ldots, f_N^t\}$. The dimension of the feature vector is then reduced to M with $M < N$ by applying the Principal Component Analysis (PCA) [118] to all the feature vectors extracted from $\mathcal{I}^{train}$.

M is the number of principal components such that a given percentage of the data variance is retained. For instance, if $M = N$, we have an exact approximation of the original data, and we say that 100% of the variance is retained. If $M = 0$, we are approximating all the data with the zero-dimensional vector, and thus 0% of the variance is retained. For this work we set a percentage of variance to 95%. After reduction, each feature vector is then normalized by using the following formula:

$$f_i^t = \frac{f_i^t - \mu_i}{\sigma_i}, \forall i \in (1, M) \tag{6.2}$$

where $\mu_i = \frac{1}{L_T} \sum_{t=1}^{L_T} f_i^t$ and $\sigma_i = \frac{1}{L_T} \sum_{t=1}^{L_T} (f_i^t - \mu_i)^2$. The normalization process makes all the feature components have zero mean and a unit variance.

The dictionary is built by grouping all the reduced feature vectors of the training set into k clusters. We adopt the kmeans algorithm that takes the number of groups or clusters k as an input parameter and outputs the best k clusters and corresponding k centroids. A centroid is the mean position of all the elements of the cluster. For each cluster we take the feature vector that is nearest to its centroid. The set of these k feature vectors composes the dictionary $\mathcal{W}$. Fig. 6.5 shows the pipeline for dictionary building. Fig. 6.6 shows examples of dictionary learned from images of the training set (anomaly free) with different patch sizes and number of clusters. The figure shows the subregions corresponding to each feature vector of the dictionary $\mathcal{W}$.

6.3.3 Learning to detect anomalies

The rationale behind the proposed method is that, in order to detect anomalies within an image, we have to estimate how much a subregion of the image is far from being

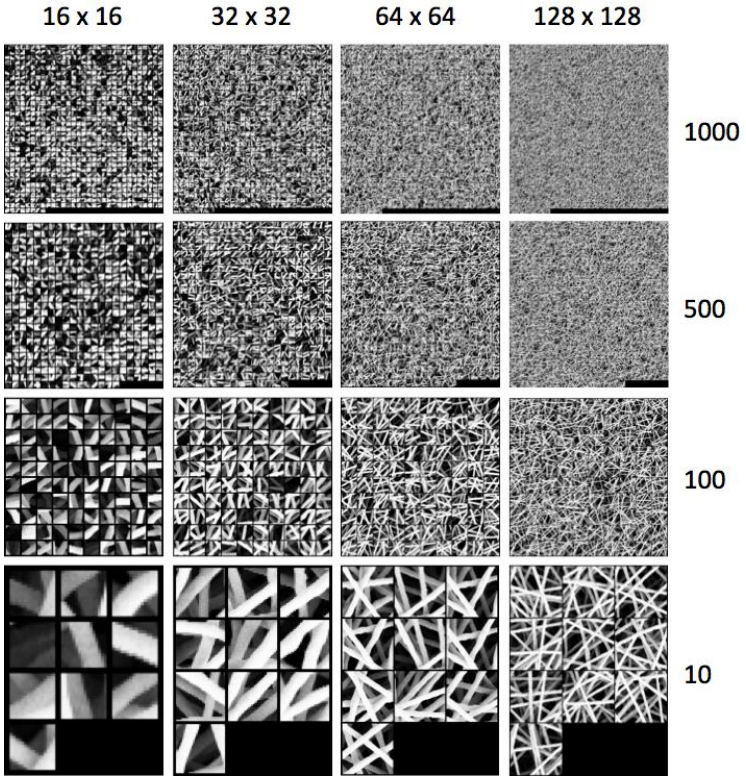


Fig. 6.6 Examples of images corresponding to the features from the dictionary. Here we show different dictionaries built with different patch sizes (16, 32, 64, 128) and number of clusters (10, 100, 500, 1000).

normal, in other words how much it is anomalous. To do so, we have to learn from one or more examples of images without anomalies the concept of "normality". We use V images from the validation set $\mathcal{I}^{val}$ that are never used for the creation of the dictionary. Each image from $\mathcal{I}^{val}$ is then processed as in the dictionary creation, that is $V_T = V \times T$ patches are extracted from V images, the feature vectors of size N are extracted from the patches and then they are reduced to size M and finally they are normalized. At the end of this operation, the average of the euclidean distances between all the feature vectors of the validation set and the m most similar subregions of the dictionary are calculated $\mathbf{d} = \{d_1, \cdots, d_{V_T}\}$. The concept of "normality" is model by the boundaries of a Gaussian function with mean and variance defined as follows: $\mu_d = \frac{1}{V_T}\sum_{t=1}^{V_T} d_t$ and $\sigma_d = \frac{1}{V_T}\sum_{t=1}^{V_T}(d_t - \mu_d)^2$. The boundaries allow to define a threshold to be used in testing time as:

$$th = \mu_d + \alpha\sigma_d \tag{6.3}$$

where α being a positive real number that modulates the size of the boundaries. The smaller will be α, the more recall-oriented will be the system. In testing time, a subregion

		Patch size			
		16	32	64	128
Layer	conv5_x (25088-dim)	2217	3083	2633	1585
	avgpool (512-dim)	151	203	175	111

Fig. 6.7 Dimension of the feature vectors after PCA reduction.

of an image is considered as anomalous if its average euclidean distance d_{test} with the m most similar subregions from the dictionary is higher than th.

6.4 Experiments

We experiment our method on SEM images of nanofibrous materials (see section A.2 for details about the dataset) and we compare it with the method proposed by Carrera et al. [25]. We experiment several versions of our method by varying the following parameters:

1. patch size: 16×16, 32×32, 64×64, 128×128. The larger is the patch the lower is the computational time and the precision in defect localization;

2. dictionary size: 10, 100, 500, 1000 number of subregions. The larger is the number the higher is the time to calculate the similarity between a test patch and the subregions of the dictionary and the better are the performance;

3. CNN layer output as feature vector: we use a ResNet-18 pre-trained on the images from ILSVRC 2015 (ImageNet Large Scale Visual Recognition Challenge) [98]. The input of the network is a RGB image of size 224×224. To adapt the input of the network to our problem we up-sample the SEM image subregion to fit the network desired size and we convert the gray-scale SEM image to a RGB one by cloning the color channels. We take the output of the `conv5_x` of the network that is a matrix $7 \times 7 \times 512$. The output is linearized to be of size 25088. Alternatively we take the output of the average pooling layer (that we name `avgpool`). The 512-dimensional feature vector is obtained by linearizing the output matrix $1 \times 1 \times 512$. All the feature vectors are L1 normalized;

4. feature dimensionality reduction: the larger is the size of the feature vector the higher is the time to calculate to calculate the similarity between a test patch and the subregions of the dictionary. In the case of PCA we consider to take the first principal components such that the retained variance of the data is about 95%. Tab. 6.7 shows, in the case of use of PCA, the reduced sizes of the feature vectors.

The smallest feature vector is obtained by combining the `avgpool` with a patch size of 128×128 while the largest is obtained by combining the `conv5_x` with a patch size of 32×32.

For all the experiments we use a stride s of 8 pixels. The kmeans clustering algorithm is performed 10 times and the best output is taken in terms of intra cluster sum of squares. The kmeans algorithm uses the euclidean distance and it is initializated through the "kmeans++" a procedure introduced by David Arthur et al. [8] to reduce the sensitivity of kmeans to the initialization seeds. The method is implemented in PyTorch (http://pytorch.org/) a python-based tool for deep learning. The experiments are launched on a Ubuntu 16.04 Personal Computer equipped with a Intel i7-4790 CPU 3.60GHz $\times$ 8, 16 GB RAM and a NVIDIA 1070 GPU.

6.4.1 Performance metrics

The performance of the proposed method is evaluated by comparing the reference anomaly map $\Omega_{\mathbf{I}}$ with the estimated anomaly map $\widetilde{\Omega_{\mathbf{I}}}$ obtained by thresholding the aggregated distance map $\Theta_{\mathbf{I}}$. At each value of the threshold corresponds an anomaly map, the smaller is the threshold, the lower will be the tolerance to which a sample is considered normal. As Carrera et al. [25] did in their work, we use two different evaluation procedures with the aim of evaluating two different aspects of the proposed method. The first one is the Receiver Operating Characteristic Curve (ROC) in which we evaluate the ability of the system to perform the per-pixel one-class classification. Specifically, we plot the true positive rate in function of the false positive rate and we use as a performance index the Area Under Curve (AUC) to evaluate the global goodness of the method. Comparing each value $\widetilde{p_i} \in \widetilde{\Omega_{\mathbf{I}}}$ with the relative ground truth value $p_i \in \Omega_{\mathbf{I}}$, we define it as:

$$true\,negative \iff \widetilde{p_i} = p_i = 0 \tag{6.4}$$

$$true\,positive \iff \widetilde{p_i} = p_i = 1 \tag{6.5}$$

$$false\,positive \iff \widetilde{p_i} = 1 \wedge p_i = 0 \tag{6.6}$$

$$false\,negative \iff \widetilde{p_i} = 0 \wedge p_i = 1 \tag{6.7}$$

The true positive rate (TPR) and the false positive rate (FPR) is defined as follows:

$$TPR = \frac{TP}{TP + FN} \tag{6.8}$$

$$FPR = \frac{FP}{FP + TN} \tag{6.9}$$

where TP,FP,TN,FN are respectively the total number of true positives, false positives, true negatives and false negatives for each map $\widetilde{\Omega_{\mathbf{I}}}$. By varying the threshold value th we obtain several values of TPR and FPR ranging from 0 to 1 and so the ROC curve. The area under curve (AUC) is the performance index used to evaluate the global goodness of the method.

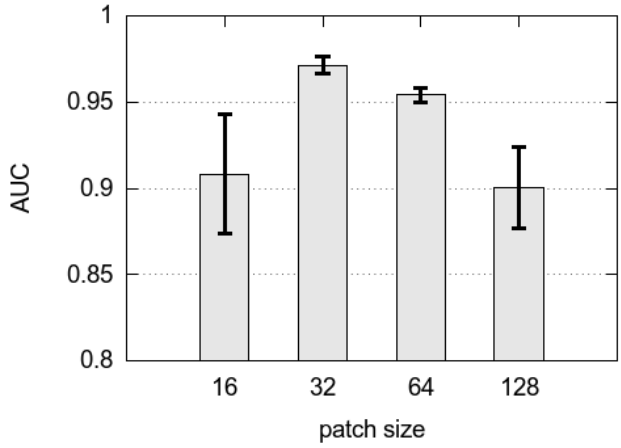
The second metric proposed by Carrera et al. [25] is the coverage percentage of defects. To this aim we choose a threshold value th such that FPR is about 5%. To compute the defect coverage percentage, we detect each defect within each reference map $\Omega_{\mathbf{I}}$ by finding each connected component $cc_j \in \Omega_{\mathbf{I}}$. Each cc_j represents a defect and for each defect we calculate the coverage factor as follows:

$$coverage\,factor_j = \frac{TP_j}{TP_j + FN_j} \tag{6.10}$$

where TP_j, FN_j are respectively the true positives and the false negatives. In other words, the coverage factor of a connected component is the number of correctly detected pixels over its area.

6.4.2 Results

Fig. 6.8 shows the AUC achieved with different variants of the proposed method. In particular Fig. 6.8(a) represents the average and standard deviation of the AUC achieved with different values of the patch size (that is 16, 32, 64 and 128) whatever is the CNN layer adopted for feature extraction (that is `conv5_x` and avgpool), whatever is the use of PCA to reduce the size of the feature vector and whatever is the number of words of the dictionary (that is 10, 100, 500, 1000). Results suggest that, in terms of AUC, the best performing variants are obtained with patch size 32 whatever are the other parameters, while the worst variants are with patch size 128 and 16. To be noticed that the best variants achieve an average AUC of about 97% while the worst variants achieve an average AUC of about 90%. As comparison, we select the method proposed by Carrera et al. [25] that is both the most performing and the most recent state of the art method on this dataset. It achieves an AUC of about 92%.



(a)

Patch size	Number of words: 10	100	500	1000
16	0.90	0.91	0.91	0.91
32	0.97	0.97	0.97	0.97
64	0.95	0.95	0.95	0.96
128	0.90	0.90	0.90	0.90

(b)

Fig. 6.8 AUC achieved with different variants of the proposed method. (a) Average and standard deviation of the AUC achieved with different patch sizes whatever is the CNN layer adopted for feature extraction, whatever is the use of PCA to reduce the size of the feature vector and whatever is the number of words of the dictionary. (b) Average of the AUC achieved with different patch sizes and number of words of the dictionary whatever is the CNN layer adopted for feature extraction and whatever is the use of PCA to reduce the size of the feature vector.

Fig. 6.8(b) shows the average of the AUC achieved with different patch sizes and number of words of the dictionary whatever is the CNN layer adopted for feature extraction and whatever is the use of PCA to reduce the size of the feature vector. Results suggest that, in terms of AUC, the best performing variants are obtained combining a patch size 32 with all the possible sizes of the dictionary. From this result is quite clear that the best solution is based on a patch size of 32×32 pixels and size of the dictionary equals to 10.

Fig. 6.9(a) and (b) show the computational time needed to process an entire image with the use of `conv5_x` and `avgpool` CNN layers respectively. The figure shows that the variants with a lower computational cost are the ones based on features extracted from the `avgpool` layer. This is related to the fact that whatever we use or not the PCA, the size of the feature vector extracted from the `conv5_x` layer is 5 times larger than the size of the feature vector extracted from the `avgpool` layer. The best variants of the proposed method are the ones that consider a patch size 32×32 and a dictionary of 10 words. The proposed method variants take about 53 seconds and 15 seconds in the case of `conv5_x` and `avgpool` respectively measured on the same machine. As argued by Carrera et al. [25] such a computational time makes it possible to monitor the production process of the nanofibers.

Patch size	Number of words: 10	100	500	1000
16	42.47	45.47	55.69	71.82
32	53.09	53.95	69.03	91.17
64	49.2	48.14	59.26	75.71
128	37.04	38.78	43.57	53.92

(a)

Patch size	Number of words: 10	100	500	1000
16	15.03	15.22	15.75	16.91
32	15.29	15.87	15.68	17.40
64	16.22	16.18	16.70	18.09
128	22.57	22.91	23.65	25.00

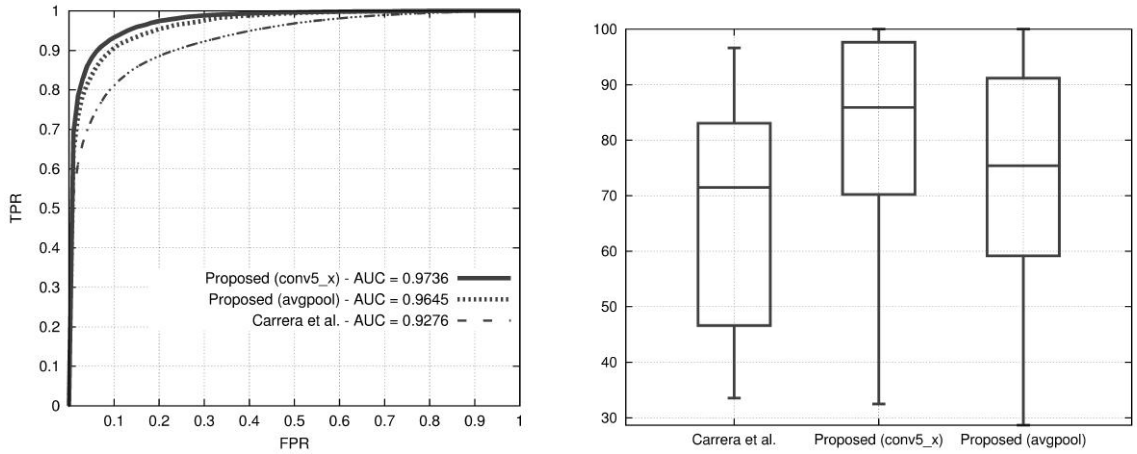
(b)

Fig. 6.9 Average time to process a test image. (a) Time needed in the case of features extracted from the `conv5_x` of the CNN. (a) Time needed in the case of features extracted from the`avgpool` of the CNN.

Fig. 6.10(a) and (b) show the ROC curves and the defect coverage box plots of the best variants of the proposed method and the state of the art. The ROC curves of both the variants of the proposed method are higher than the state of the art and the AUC of both variants is about 5% higher than the state of the art. It is quite interesting to note that the variant with the `avgpool` layer achieves almost the same AUC than the one with the `conv5_x` while having a computational time that is about 3 time less than the computational time of the `conv5_x`-based one.

The box plots of Fig. 6.10(b) show that the state of the art solution achieves an average value of defect coverage that is quite similar to the worst value obtained by the best variant of the proposed method, that covers at least the 50% of the anomalies for more than 85% of their area.

Fig. 6.11 shows some defect detections and localizations obtained with a variant of the proposed method that use `conv5_x` layer, patch size 32, PCA for dimensionality reduction and a dictionary of size 10. The images are close-ups of fine- and coarse-grained defects. True positives are green colored, false positives are red colored while false negatives are blue colored. It is quite evident that the proposed method is quite accurate to detect coarse grained defects and it should be further improved to detect medium- and fine-grained defects.



(a) (b)

Fig. 6.10 Results from the two variants of the proposed method, one with `conv5_x` and the other with `avgpool`, and comparison with the method proposed by Carrera et al. [25]. Both the variants consider the PCA to reduce the feature vector, a patch size of 32 pixels and a dictionary of size 10. (a) ROC curves. For each ROC curve, the corresponding AUC values are in the legend. (b) Box-plots reporting the distribution of the defect coverage obtained at a fixed FPR = 5%.

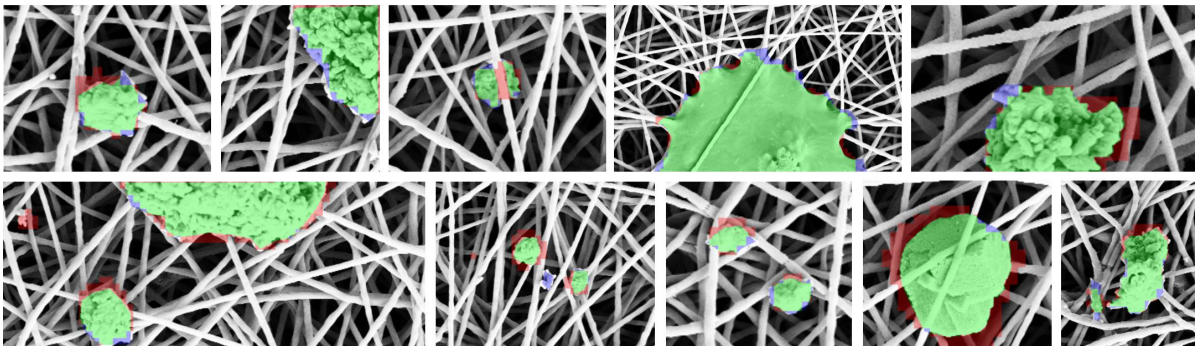


Fig. 6.11 Closeup of the anomalies found by the proposed method. True positives, false positives, false negatives are showed respectively as green,red and blue color. For visualization purpose, the images are slightly cropped and scaled to focus on fine- and coarse-grained anomalies.

Chapter 7

Generative Methods For Anomaly Removal

"Men are disturbed not by things, but by the view which they take of them."

Epictetus

7.1 Introduction

The detection of *spatially distributed* anomalies, in contrast to *highly localized* anomalies, requires a different approach. Specifically, the key idea to detect them is firstly to reconstruct the image. The proposed method is of type *generative* (see figure 7.1 for reference). In the context of this chapter, we consider *spatially distributed* anomalies as local color effects that usually can be obtained with a filtering technique. Therefore, we concentrate on the task of removing photographic filters from normal images. Starting from this moment, with the term *unfiltering* we refer at the same time with the identification of an anomaly and to its removal by projecting the input image in the manifold of normality as well as to the recovery of a filtered image to its origial version.

7.2 Proposed method

We propose here a method for the automatic removal of photographic filters. The method takes as input a color image that has been possibly processed with a photographic filter, and produces as output a color image representing the same content, but with the style modified to reproduce the appearance of a "natural", unfiltered image. Note that no knowledge is required about which filter (if any) needs to be removed.

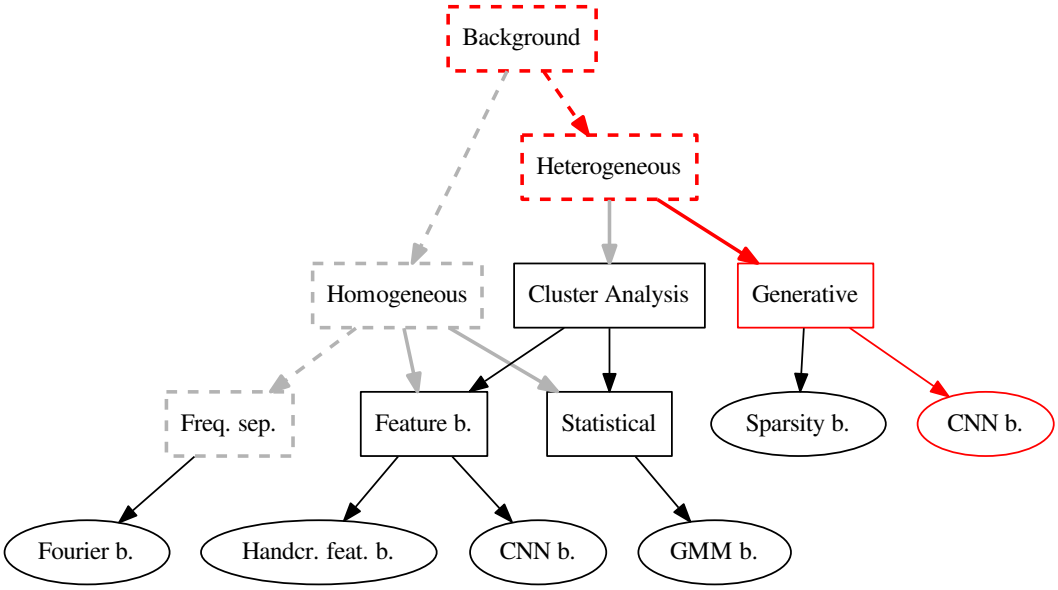


Fig. 7.1 The method presented in this chapter is able to work with heterogeneous backgrounds.

Not all the editing operations involved in the computations of the filters are invertible. In fact, some of them cause a loss of information, making unfiltering an ill-posed problem. For instance, to an image processed by a filter that includes a conversion to gray-level (such as *Gotham* or *Inkwell*) correspond many plausible unfiltered images. However, as we will show in Section 7.3, our method is often able to guess a reasonable recovery of the missing information by inferring it from the semantic content of the input image (for instance, by recognizing the sky in a gray-level image and by coloring it in blue).

Many image-to-image deep learning models have been recently proposed [60, 128, 56]. Their results are often remarkable, but they come at the price of a high computational cost. In fact, all the information in the input image that is required to generate the output has to be preserved through all the layers, either by using large intermediate representations [52] or by using skip connections [96]. Applied to the problem at hand, this fact implies that all the fine details that are not affected by the photographic filters need to be preserved from input to output. To address this issue, we diverged from the popular image-to-image approach: instead of directly estimating the pixel values of the original unfiltered image, our model estimates the parameters of a set of

local transformations that, when applied to the input image, approximate the desired output. Note that a single global transformation would not be suitable, since many filters (e.g. *Amaro*) are spatially varying. However, a small number of local transformations may be enough to reverse the photographic filters considered keeping manageable, at the same time, the complexity of the model and the number of parameters that needs to be learned. We chose to work with polynomial transformations, since they have been demonstrated to be very effective for color processing[62, 18, 19]. More precisely, we decided to use a grid of $T \times T$ polynomials that are then bilinearly upsampled to produce a per-pixel transformation (this can be done since the set of polynomials is closed under linear combinations).

In addition to the relatively low computational requirements, our approach also has the advantage of keeping the training procedure simple. In fact, one of the problems of image-to-image methods is that they require a complex loss function (often in the form of an adversarial network) to avoid losing the details of the image [60, 56]. Our model, instead, can be trained simply by minimizing the mean squared error between the desired and the actual output since details are preserved by the local transformations.

Given an input image of size 256×256, the method works as follows: first for each pixel x the (x_R, x_G, x_B) components are projected into the monomial basis:

$$(1, x_R, x_G, x_B, x_R x_G, x_R x_B, x_G x_B, x_R^2, x_G^2, x_B^2, \ldots, x_R^D, x_G^D, x_B^D) \tag{7.1}$$

for a set degree D. The result of this operation, that we call 'polynomial expansion', is an image of $\binom{D+3}{D}$ channels. After that the method applies a sequence of five convolutional blocks each one including a 3×3 convolution with 200 output channels, preceded by batch normalization and followed by the ReLU activation function. Convolutions are applied with a stride of two in both spatial dimensions to progressively reduce the size of the image. The resulting $7 \times 7 \times 200$ image is flattened into a vector that is processed by a sequence of two linear layers (followed by a ReLU) producing a vector of 2000 components. A further linear transformation produces the coefficients of $3 \times T \times T$ polynomials of degree D (T^2 polynomials for each of the three color channels, each one defined by $\binom{D+3}{D}$ coefficients). Bilinear upsampling is then applied to smoothly interpolate the $T \times T$ transformations over the 256×256 input in such a way that each pixel has is own triplet of polynomials (for the R, G and B channels). Finally the polynomial expansion of the input image is multiplied by the interpolated coefficients yielding to the unfiltered image. Note that, thanks to the fully connected linear layers, the coefficients of each local transformation depends on the whole image. This facilitates the modeling of global

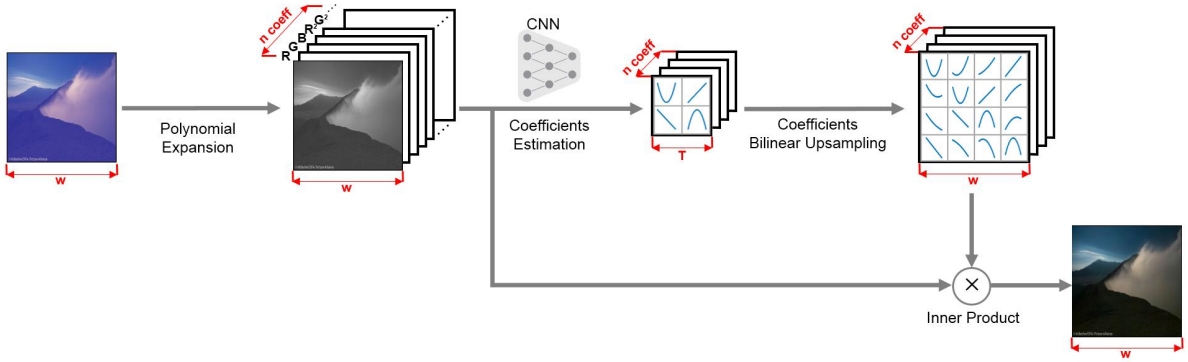


Fig. 7.2 Pipeline of the unfiltering process. The input image is unfiltered through a set of local polynomial color transformations whose coefficients are estimated by the CNN.

corrections, when appropriate. Note also that the bilinear upsampling of the polynomials makes it easy to preserve the details in the input image.

We experimented with polynomial transformations up to the third degree, implying the use of to the following polynomial expansions:

$$\begin{aligned} E_1(x) =& (1, x_R, x_G, x_B), \quad (D=1), \\ E_2(x) =& (1, x_R, x_G, x_B, x_R x_G, x_R x_B, x_G x_B, x_R^2, x_G^2, x_B^2), \quad (D=2), \\ E_3(x) =& (1, x_R, x_G, x_B, x_R x_G, x_R x_B, x_G x_B, x_R^2, x_G^2, x_B^2, \\ & x_R x_G x_B, x_R^2 x_G, x_R^2 x_B, x_R x_G^2, x_R x_B^2, x_G^2 x_B, x_G x_B^2, x_R^3, x_G^3, x_B^3), \quad (D=3). \end{aligned} \tag{7.2}$$

The bilinear interpolation produces, for each pixel of the input image, a set of coefficients $\{k_{ijk}^c\}$ with $i+j+k \leq D, c \in \{R,G,B\}$. The components (y_R, y_G, y_B) of a pixel in the output image are finally computed as in the following expression:

$$y_c = \sum_{i+j+k \leq D} k_{ijk}^c x_R^i x_G^j x_B^k, \quad c \in \{R,G,B\}, \tag{7.3}$$

that can be casted as an inner product between the set of coefficients (suitably encoded as a vector) and the polynomial expansion in Equation (7.2).

The whole method is summarized in Figure 7.2 and in Table 7.1.

We used stochastic gradient descent with mini-batches to train the CNN, by minimizing the mean squared error (MSE, Equation 7.4) between the output pixels y and the corresponding ground truth $\hat{y}$:

$$MSE = \frac{1}{3 \times 256^2} \sum_{i=1}^{256} \sum_{j=1}^{256} \sum_{c \in \{R,G,B\}} \left(y_c(i,j) - \hat{y}_c(i,j)\right)^2. \tag{7.4}$$

Table 7.1 Structure of the convolutional neural network. D denotes the degree of the polynomial transformations while T^2 is their number of local transformations. All convolutional layers have filters of dimension 3×3 and stride 2.

Stage	Operation	Output size
Pre-processing	Input	$256 \times 256 \times 3$
	Polynomial Expansion	$256 \times 256 \times \binom{D+3}{D}$
Conv. Network	Batch Norm. + Conv. + ReLU	$127 \times 127 \times 200$
	Batch Norm. + Conv. + ReLU	$63 \times 63 \times 200$
	Batch Norm. + Conv. + ReLU	$31 \times 31 \times 200$
	Batch Norm. + Conv. + ReLU	$15 \times 15 \times 200$
	Batch Norm. + Conv. + ReLU	$7 \times 7 \times 200$
	Linear + ReLU	2000
	Linear + ReLU	2000
	Linear	$T \times T \times 3 \times \binom{D+3}{D}$
Post-processing	Bilinear Upsampling	$256 \times 256 \times 3 \times \binom{D+3}{D}$
	Polynomial Transformation	$256 \times 256 \times 3$

7.3 Experimental results

To assess the effectiveness of the proposed method, we run various experiments where test images are processed and the result is compared against the original image, before the application of photographic filters. We evaluated three aspects:

- faithfulness of the result of the unfiltering process with respect to the ground truth original image, measured with objective error metrics;
- 'naturalness' of the result, measured by a neural network trained to identify filtered images;
- improvement in recognizability of the unfiltered content, measured by a network trained to classify the image content.

7.3.1 Error metrics

Four different objective metrics are used to evaluate the quality of the recovered images. The first two are the simplest and most widely used full-reference quality metrics: the former is the Mean Squared Error (MSE). Given an image I and its recovered version K,

MSE is defined as:

$$MSE = \frac{1}{m\,n}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i,j) - K(i,j)]^2. \tag{7.5}$$

This metric is used since it is the one used as loss function in the training of our CNN. The latter is the Peak Signal-to-Noise Ratio (PSNR) and is related to MSE:

$$PSNR = 20 \cdot \log_{10}\left(\frac{MAX_I}{\sqrt{MSE}}\right), \tag{7.6}$$

where MAX_I is the maximum possible pixel value of the image.

The third error metric considered is the Structural Similarity (SSIM) index [116]. It exploits the assumption that human visual perception is highly adapted for extracting structural information from a scene, and is an alternative complementary framework for quality assessment based on the degradation of structural information. In particular, it is used here to assess if the filter removal process degrades the structural information present in the original image. The general form of SSIM combines information from the luminance $l(I,K)$, the contrast $c(I,K)$ and structure $s(I,K)$ as follows:

$$SSIM(I,K) = l(I,K)^\alpha \cdot c(I,K)^\beta \cdot s(I,K)^\gamma \tag{7.7}$$

where $\alpha > 0$, $\beta > 0$ and $\gamma > 0$ are parameters used to adjust the relative importance of the three components (we used the values $\alpha = \beta = \gamma = 1$).

The fourth error metric considered is the spatial extension of CIELAB (S-CIELAB) that has been specifically designed for measuring reproduction errors of color images [123].

7.3.2 Performance varying the degree and number of polynomials

In this section we show the performance we obtained in terms of the four error metrics considered in the previous section. Our method depends on two parameters: the degree D of the polynomial transformations and their number T^2.

Table 7.2 reports the results obtained by changing the degree of the polynomial used for the recovery, as well as the number of transformations considered. The number of transformations we considered are $32 \times 32 = 1024$, $16 \times 16 = 256$, $8 \times 8 = 46$, $4 \times 4 = 16$, $2 \times 2 = 4$ and $1 \times 1 = 1$ (i.e. a global transformation of the whole image). From the results it can be noticed that the performance obtained with 8×8 transformations or more are almost equivalent and that they start to decrease when fewer transformations

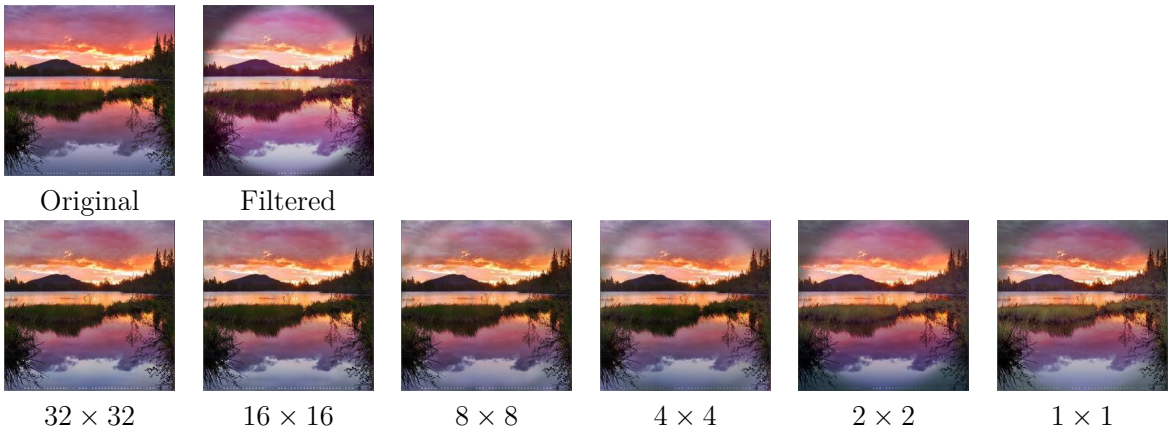


Fig. 7.3 Example of removal of a spatially varying filter (*Amaro*) with a different number of polynomial transformations. As the number of transformation decreases, the quality of the restoration gets worse.

are used. This behavior is consistent across all the four error metrics considered. A visual example is reported in Figure 7.3, where the results obtained using the different number of third degree polynomial transformations considered are reported. Fixing the number of transformations, it is possible to notice how the results improve for all the four error metrics considered at the increase of the polynomial degree. The same conclusion can be drawn also by fixing the number of parameters required, i.e. looking at the results in Table 7.2 taking into account the color coded background. Therefore, for the next experiments we considered only the configuration of the proposed method with 32×32 polynomial transformations of third-degree.

7.3.3 Comparison with the state of the art

In this section we compare the results of the proposed method with those of other algorithms from the state of the art. The first one is the Gray World (GW) [22], which is a global color correction algorithm. It is based on the assumption that the average reflectance in a natural scene is gray, and it balances the image channels in order to make their average value match. The second one is the Retinex algorithm [71], which is able to deal with non-uniform illumination by assuming that an abrupt change in chromaticity is caused by a change in reflectance properties. The third method is the image to image translation architecture recently proposed by Isola et al. [60], that we used in two variants: the former obtained with the default setup recommended by the authors and the latter obtained by disabling the GAN loss during training, so that the method becomes a regression with a per-pixel L_1 loss. The comparison is reported in

Polynomial degree D	Number of transformations $T \times T$					
	32×32	16×16	8×8	4×4	2×2	1×1
$D = 1$	30.70	30.57	30.47	29.47	27.83	27.53
$D = 2$	31.60	31.49	31.41	30.43	28.64	28.33
$D = 3$	32.01	31.99	31.92	30.83	28.86	28.50

a) PSNR (higher is better)

Polynomial degree D	Number of transformations $T \times T$					
	32×32	16×16	8×8	4×4	2×2	1×1
$D = 1$	0.9343	0.9332	0.9327	0.9288	0.9141	0.9123
$D = 2$	0.9403	0.9400	0.9394	0.9361	0.9203	0.9210
$D = 3$	0.9435	0.9435	0.9431	0.9395	0.9252	0.9257

b) SSIM (higher is better)

Polynomial degree D	Number of transformations $T \times T$					
	32×32	16×16	8×8	4×4	2×2	1×1
$D = 1$	78.94	81.15	82.03	100.19	155.86	178.72
$D = 2$	66.13	67.47	68.58	83.67	135.56	148.31
$D = 3$	61.61	61.34	62.37	78.08	129.43	141.85

c) MSE (lower is better)

Polynomial degree D	Number of transformations $T \times T$					
	32×32	16×16	8×8	4×4	2×2	1×1
$D = 1$	5.14	5.33	5.22	5.74	5.93	6.14
$D = 2$	4.71	4.82	4.80	4.98	5.72	5.79
$D = 3$	4.72	4.75	4.71	5.11	5.64	5.66

d) S-CIELAB (lower is better)

e)

Table 7.2 Comparison of the performance of the proposed method varying the degree of the polynomial and the number of transformations used. Results are assessed using the four error metrics considered: PSNR (a), SSIM (b), MSE (c) and S-CIELAB (d). The background is color-coded with the legend reported in (e), to represent the total number of parameters used by the transformations.

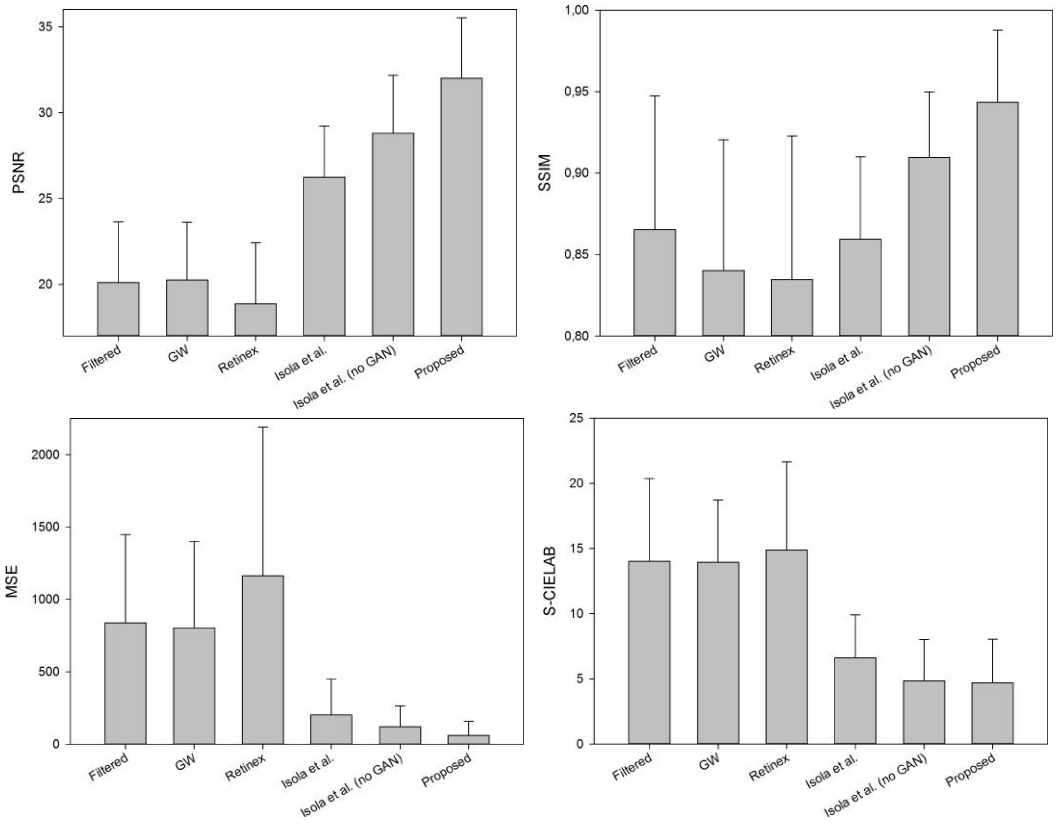


Fig. 7.4 Comparison of the proposed method with the state of the art for the four error metrics considered: PSNR, SSIM, MSE and S-CIELAB. For the former two metrics (reported in the top row) the higher the better, for the latter two (reported in the bottom row) the lower the better.

Figure 7.4 in terms of all the four error metrics considered. We can see from the plots that our method significantly outperforms the compared methods.

Some visual examples of images recovered with all the compared methods are reported in Figure 7.5. Figure 7.6 shows a couple of enlargements of portions of the same image recovered with our method compared with the two variants of Isola et al. [60], from which it is possible to notice how our method is able to preserve also the finest details.

As a further analysis we report in Figure 7.7 the break down of the results with respect to the different filters considered. From the results it is possible to see that the performance of the different methods tested is consistent across the filters considered with very few ranking inversions. Focusing on the proposed method, we can notice that the worst results across the four error metric considered are obtained on those filters characterized by a heavy information loss, i.e. Gotham, Inkwell and Willow, that are all black & white filters. This is particularly evident looking at the plot of the S-CIELAB metric, where it can be seen that the error on these three filters is almost the double of

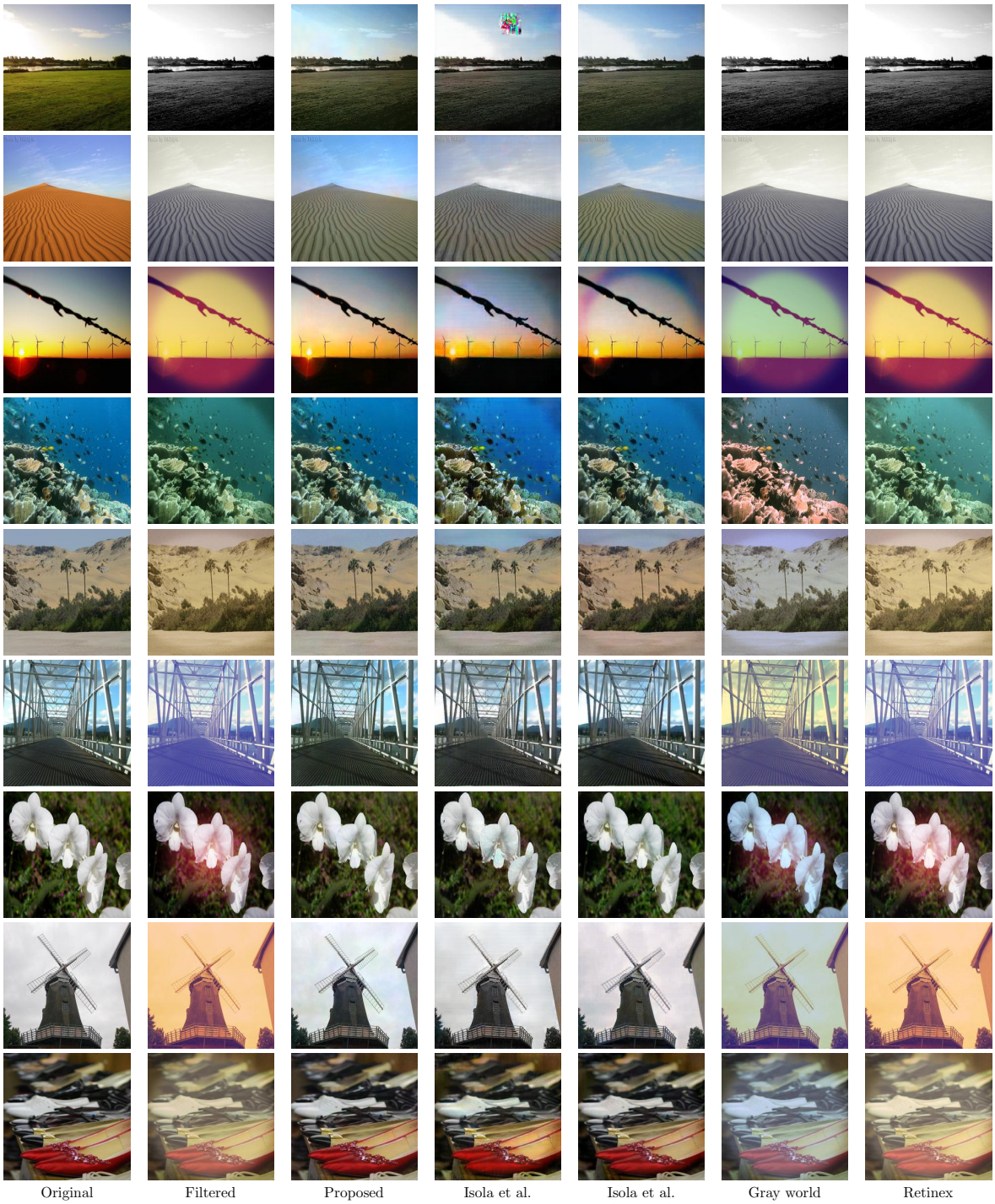


Fig. 7.5 Examples of recovery by our proposed method, gray world, Retinex, and by two variants of the method by Isola et al. [60]

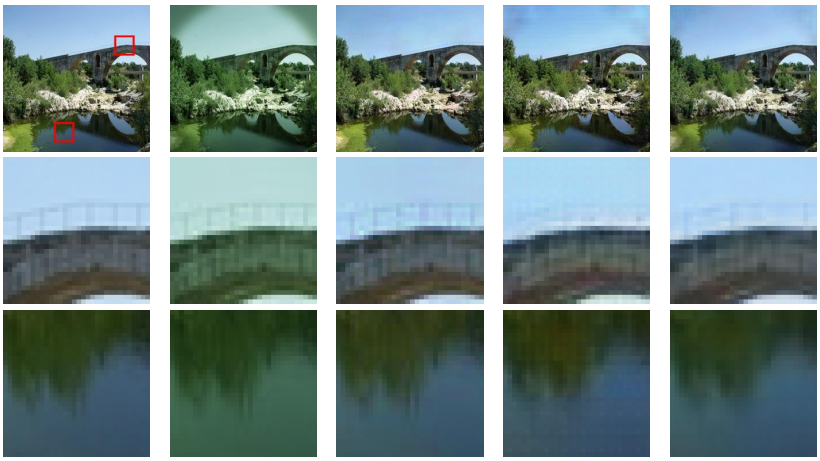


Fig. 7.6 Qualitative comparison of the details produced by the proposed method and by the method by Isola et al., with and without the GAN term in the loss function.

the others. Nevertheless, this error is lower than that of the filtered image, which means that the proposed method also performs a form of colorization.

7.3.4 Filter classification

The filter removal procedure described here produces an output image that most of the times looks more natural than its filtered version. To partially quantify this aspect, we measured how often this happens from the point of view of a convolutional network trained to recognize the application of photographic filters. More in detail, we considered the network that Bianco et al. [15] designed to recognize the same 22 photographic filters used in this work. The accuracy of that network was very high (about 98.9%), as it can be seen in the confusion matrix summarizing the results of classification for the test images (Table 7.3). All the 22 filters are correctly recognized in at least 97% of the times and original images are recognized as such in 91.5% of the cases.

We assessed the same network on the test images after removing the filters with the proposed method. The results, reported as confusion matrix in Table 7.4 are very different. The classification accuracy dropped to about 5.39%, which is slightly better than random guessing. In the great majority of cases the network classified the recovered images as original, i.e. without any photographic filter applied. This happens, on average, 89.4% of the times. It seems that the recovery process is very good in canceling out those characteristic properties that make the filters recognizable for the neural network.

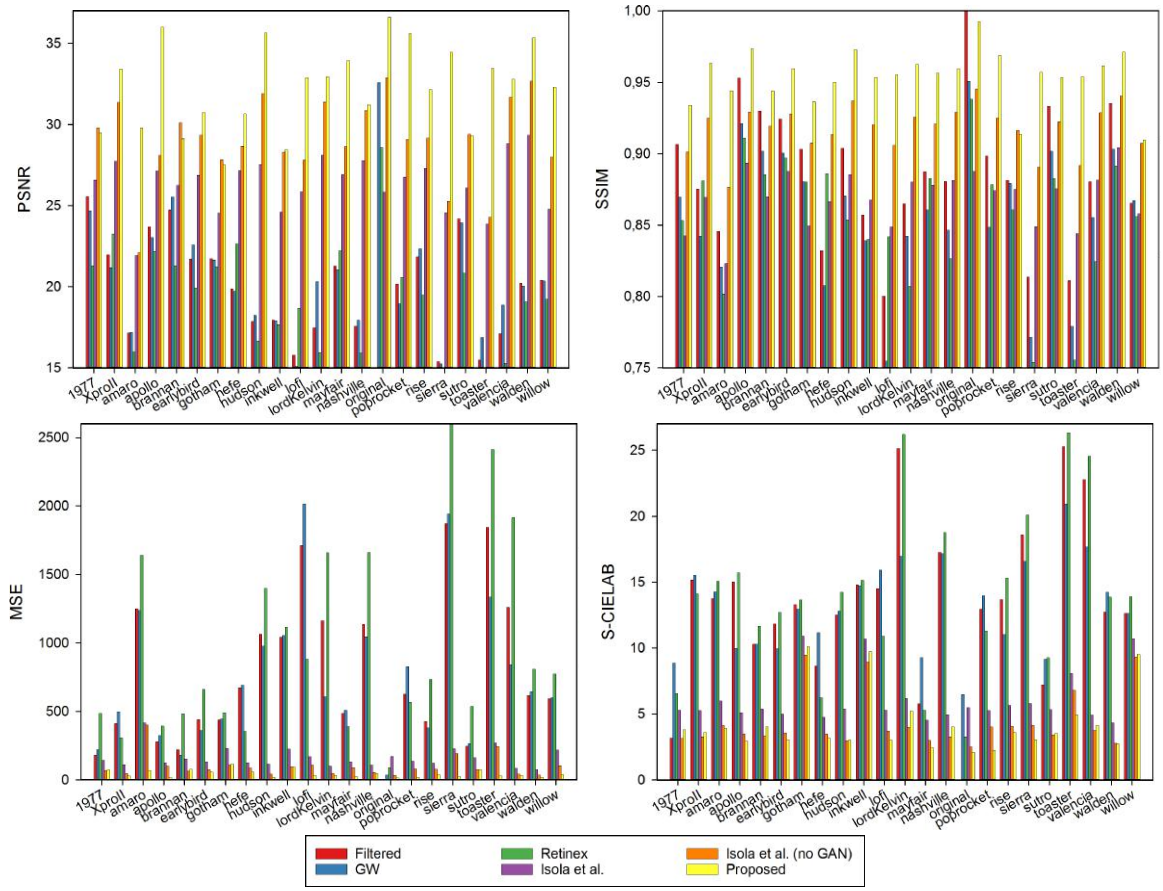


Fig. 7.7 Detailed comparison of the proposed method with the state of the art in removing each of the 23 filters considered. Results are reported using the four error metrics considered: for those in the top row the higher the better, while for those in the bottom row the lower the better.

Table 7.3 Confusion matrix summarizing the output of the filter classification network proposed by Bianco et al.[15], obtained on the test images transformed by photographic filters. For each filter, the most common prediction is reported in bold.

	Original	1977	Amaro	Apollo	Brannan	Earlybird	Gotham	Hefe	Hudson	Inkwell	Lofi	Lord-K.	Mayfair	Nashville	Poprocket	Rise	Sierra	Sutro	Toaster	Valencia	Walden	Willow	X-pro II
Orig	**91.5**	0.6	0.1	0.2	0.3	0.3	0.0	1.9	1.3	0.1	0.2	-	1.2	0.1	0.0	0.9	-	0.4	-	0.0	0.2	0.1	0.3
1977	0.1	**99.2**	-	-	0.2	-	-	-	0.2	-	-	-	-	0.1	-	-	-	0.1	-	0.1	0.2	-	-
Amar	0.0	-	**99.9**	0.0	-	-	-	-	-	-	-	-	-	-	-	-	0.0	-	-	-	-	-	-
Apol	0.5	-	0.0	**99.2**	-	-	-	0.0	0.0	-	0.0	-	-	-	-	0.2	-	-	-	-	-	-	-
Bran	0.1	0.3	-	-	**99.0**	-	0.0	-	0.1	-	-	0.0	0.0	0.0	-	-	-	0.1	-	0.1	0.1	-	0.1
Earl	0.3	0.1	-	0.1	0.1	**99.0**	-	0.1	-	-	0.1	-	-	-	-	0.1	0.1	-	-	0.0	0.1	-	-
Goth	-	-	-	-	-	-	**100.0**	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Hefe	0.8	-	-	0.1	-	-	-	**98.6**	-	0.1	0.2	-	0.1	-	0.0	0.1	-	0.0	-	0.0	-	-	-
Huds	0.7	0.2	0.0	-	0.2	-	-	-	**98.4**	-	-	-	-	0.1	0.1	0.0	0.0	0.0	-	0.1	0.2	-	0.1
Inkw	0.0	-	-	-	-	-	-	0.0	-	**97.9**	-	-	-	-	-	-	-	-	-	-	-	2.1	-
Lofi	0.1	-	-	0.0	-	-	-	0.0	-	-	**99.9**	-	-	-	0.0	-	-	-	-	0.0	-	-	-
Lord	-	-	-	-	-	-	-	-	-	-	-	**99.9**	-	0.1	-	-	0.0	-	0.0	0.0	-	-	-
Mayf	0.9	-	-	0.0	-	0.1	-	0.1	0.1	0.1	0.0	-	**98.5**	-	0.0	0.1	-	-	-	0.0	-	0.0	0.1
Nash	-	-	-	-	-	-	-	-	-	-	-	0.1	-	**99.8**	-	-	-	-	-	0.0	0.1	-	0.1
Popr	0.0	-	-	-	-	-	-	-	0.1	-	-	-	-	-	**99.9**	0.0	-	-	-	-	0.0	-	-
Rise	0.6	-	-	0.5	0.1	0.1	-	-	0.1	-	-	-	0.1	-	0.0	**98.3**	-	0.1	-	0.0	-	-	-
Sier	-	-	-	-	-	-	-	-	0.0	-	-	-	-	-	-	-	**100.0**	-	-	0.0	-	-	-
Sutr	0.1	0.1	-	-	0.1	-	-	0.0	0.0	-	0.1	-	-	0.0	-	-	-	**99.5**	-	0.1	0.1	-	-
Toas	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	**100.0**	-	-	-	-
Vale	0.0	-	-	-	-	-	-	-	-	-	-	-	-	0.1	-	-	-	-	-	**99.9**	-	-	-
Wald	-	0.1	-	-	0.1	-	-	-	0.1	-	-	0.1	-	0.5	-	-	-	0.1	-	0.0	**99.1**	-	0.1
Will	0.1	-	-	-	-	-	-	-	-	1.9	-	-	-	-	-	-	-	-	-	-	-	**98.0**	-
X-pr	0.0	-	-	0.0	0.1	-	-	-	-	-	-	-	-	0.1	-	-	-	-	-	0.0	0.1	-	**99.6**

Table 7.4 Confusion matrix summarizing the output of the filter classification network proposed by Bianco et al.[15], obtained on the test images restored by the proposed method. For each filter, the most common prediction is reported in bold.

	Original	1977	Amaro	Apollo	Brannan	Earlybird	Gotham	Hefe	Hudson	Inkwell	Lofi	Lord-K.	Mayfair	Nashville	Poprocket	Rise	Sierra	Sutro	Toaster	Valencia	Walden	Willow	X-pro II
Orig	**96.0**	0.2	0.1	0.3	-	0.4	-	1.2	0.1	-	0.1	-	0.9	-	-	0.4	-	0.1	-	0.0	0.0	-	0.1
1977	**81.4**	2.4	0.0	0.2	0.3	0.1	-	0.1	10.7	-	0.5	-	0.2	-	0.0	1.7	0.0	2.2	-	0.0	0.1	-	0.1
Amar	**90.8**	0.6	0.2	0.2	0.2	0.4	-	2.2	2.2	-	0.3	-	1.1	-	0.0	0.6	0.0	0.7	-	-	0.2	-	0.1
Apol	**90.9**	0.7	0.1	1.8	0.2	0.6	-	1.3	1.5	-	0.1	-	0.9	0.1	-	1.2	-	0.3	-	0.0	0.1	-	0.2
Bran	**88.3**	1.0	0.1	0.2	1.0	0.2	0.0	1.2	2.4	-	0.6	-	0.7	-	-	1.3	-	2.8	-	0.0	0.0	-	0.3
Earl	**89.1**	0.8	-	0.4	1.0	1.4	0.0	1.9	0.9	-	0.2	-	0.9	0.0	0.0	1.3	-	1.6	-	0.0	0.0	-	0.2
Goth	**88.7**	1.3	0.0	0.7	0.1	0.4	0.1	1.1	0.7	0.0	0.1	-	0.7	-	-	5.4	-	0.8	-	-	-	0.0	-
Hefe	**88.3**	0.4	0.0	0.2	0.4	0.1	0.1	4.4	1.4	-	0.4	-	1.0	-	-	2.2	-	0.9	-	-	0.1	-	0.1
Huds	**91.0**	0.6	0.0	0.2	0.2	0.6	0.0	1.7	1.2	-	0.5	0.0	0.9	0.0	-	0.9	-	1.8	-	0.0	0.1	-	0.2
Inkw	**88.4**	0.7	-	0.5	0.1	0.9	0.1	0.6	0.0	0.1	0.2	-	0.8	-	-	6.4	-	1.0	-	-	-	0.1	-
Lofi	**91.1**	0.8	-	0.2	0.3	0.1	0.0	1.4	2.0	-	0.9	-	1.0	0.1	-	1.1	-	0.5	-	-	0.2	-	0.2
Lord	**91.0**	0.6	0.1	2.3	0.1	0.4	-	1.7	0.6	-	0.1	-	1.4	-	0.0	0.7	-	0.8	0.0	-	-	-	0.2
Mayf	**89.8**	0.7	0.0	0.2	0.2	0.2	0.1	0.9	1.2	-	0.4	-	3.6	0.0	0.1	1.4	-	0.7	-	0.1	0.2	0.0	0.2
Nash	**88.8**	1.8	-	0.3	0.7	0.8	-	1.8	0.3	-	0.2	-	0.9	-	-	0.8	0.1	2.6	-	-	0.5	-	0.5
Popr	**90.2**	0.8	0.1	0.4	0.4	0.5	0.0	1.8	1.8	-	0.3	-	1.1	-	0.3	1.0	-	0.7	-	0.0	0.2	-	0.3
Rise	**91.1**	0.8	0.1	0.7	0.2	0.2	-	1.3	1.5	-	0.2	-	1.1	0.0	0.0	2.2	-	0.5	-	-	0.1	-	0.1
Sier	**91.5**	1.0	0.1	0.3	0.6	0.4	-	1.1	2.6	-	0.3	0.0	0.8	0.0	0.0	0.7	0.1	0.2	-	-	0.2	-	0.2
Sutr	**83.0**	1.1	0.1	0.4	0.6	0.2	0.1	2.3	1.3	-	0.5	-	1.2	-	-	2.5	-	6.8	-	-	0.0	-	0.1
Toas	**91.1**	0.7	0.1	0.2	0.6	1.2	-	0.7	2.8	-	0.4	-	0.7	0.1	0.0	0.8	-	0.8	-	-	0.0	-	-
Vale	**88.6**	1.2	-	1.8	0.2	0.1	0.1	1.7	1.4	-	0.2	0.0	0.9	0.1	0.0	2.0	-	1.0	-	0.0	0.7	-	0.1
Wald	**92.2**	0.8	0.1	0.4	0.4	0.5	-	1.4	0.9	-	0.2	-	1.8	-	-	0.5	0.0	0.2	-	0.0	0.5	-	0.2
Will	**86.2**	0.9	-	0.2	0.1	2.8	0.0	1.5	0.2	0.0	0.1	-	0.3	-	-	4.5	-	3.0	-	-	-	0.0	-
X-pr	**89.8**	0.5	0.0	0.3	0.4	0.1	-	1.0	0.9	-	0.4	-	1.0	0.0	-	2.8	-	1.6	-	0.0	0.2	-	0.9

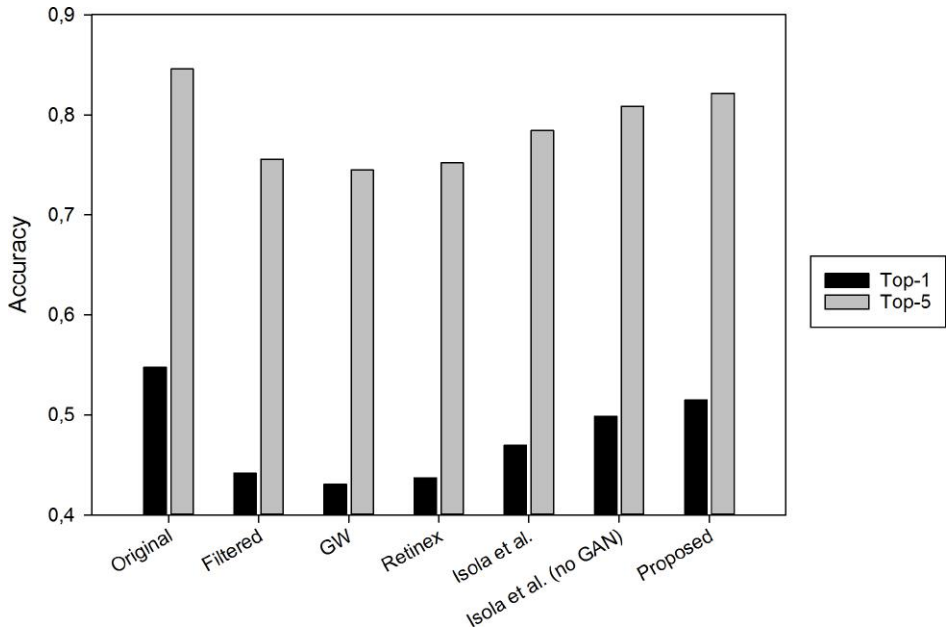


Fig. 7.8 Classification accuracy of a CNN trained to recognize the semantic labels in the Places-205 dataset.

7.3.5 Semantic classification on Places-205 dataset

As a further experiment, we assessed the classification performance of a CNN trained to identify the semantic classes defined in the Places-205 dataset. The CNN architecture used is AlexNet [68] and the trained model can be downloaded on the website of Places-205. The classification accuracy is measured on the original images (on which no filter was applied), on the filtered images, on the images corrected with the other methods included in the evaluation, and on those corrected with the proposed method. From Figure 7.8 it is possible to see that, as expected, the best classification result is obtained on the original images. When the filters are applied, classification accuracy drops by almost 10% for the top-1 result. Correcting the images with GW and Retinex does not improve this result, but it actually lowers the accuracy by almost a further 1%. The proposed method instead, is able to reduce the gap with respect to the results obtained on the original images, increasing the accuracy by almost 7% with respect to the results obtained on the filtered images.

7.3.6 Analysis of the learned features

Finally, in order to better understand the behavior of the network, we conducted an analysis on the features computed by the second fully-connected linear layer. More

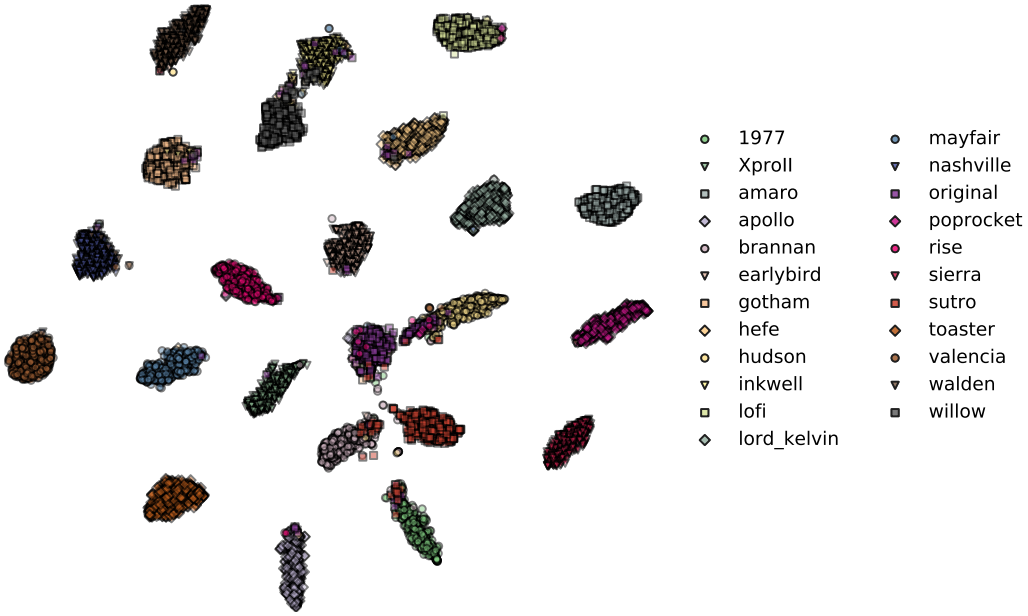


Fig. 7.9 Output of the t-SNE data visualization method applied to a set of 10 000 feature vectors computed by the second fully-connected layer. The projected data points are displayed according to the corresponding photographic filter, even though this information was not available to t-SNE.

in detail, we randomly sampled 10 000 images from the test set and used the t-SNE method [80] to project the 2000-dimensional feature vectors produced by that layer onto a plane. The projection computed by t-SNE preserves the similarity among the feature vectors and it is determined in a completely unsupervised way, without any knowledge about the photographic filter applied to the image. From the results depicted in Figure 7.9, it is clear how the method identified 23 clusters, one for each photographic filter. This fact suggests that the main purpose of the second fully-connected layer is to recognize which filter has been applied to the input image, an information that allows to effectively select the restoration strategy that is implemented in the following layer.

Part V

Epilogue

Chapter 8

Conclusions

In this thesis it has been done an in-depth study of anomaly detection for industrial quality inspection, ensured through the analysis of images depicting the product under inspection. In the introduction (part I) it has been done an extensive study on the subject to introduce the reader and to propose a pipeline for automatic anomaly detection. This pipeline is composed by two steps: 1) the enhancement of the input images for highlighting defects; 2) the detection of the anomalies.

The first step has been addressed with the use of a global color transformation able to remove undesired light effects and to enhance the contrast. This transformation is inferred through the use of SpliNet, a new CNN-based method presented in this thesis in part III, that is able to enhance the input images by inferring the parameters of a set of splines. The method allows to retouch images preserving their content and without introducing any artifacts. The results obtained demonstrate that the presented approach allows to reproduce with great fidelity the retouching styles of the individual experts. Moreover, this method performs favorably with respect to other methods that have been recently proposed in the literature. In this context, it is also presented an extension of the system to adapt the preprocessing to a new retouching style, without having to retrain the system. In future it is planned to extend this step by adopting a larger dataset, that includes more diverse range of subjects and image conditions. Furthermore, it is also planned to augment the proposed method by injecting additional semantic information obtained by suitably trained neural networks.

In the context of anomaly detection, two methods have been presented in part IV. The first one (chapter 6) is a feature-based method that models normality by learning a dictionary and uses it in test time to determine the degree of abnormality of an inquiry image of the product under inspection. This method has been benchmarked on the detection and localization of anomalies in Scanning Electron Microscope images of

nanofibrous material, where it has demonstrated great flexibility and high accuracy even on very small datasets (in the order of five images). This method outperformed existing methods of about 5% reaching an area under curve of about 97% and represents so far the state of the art. Future investigations for extending this method could include the processing of the input images at different scales in order to be more accurate in the detection of small anomalies.

The second proposed algorithm (chapter 7) in the context of anomaly detection is a generative method able to restore the input, creating an anomaly-free version of the inquiry image. Subject of this study is the detection of *spatially distributed anomalies*, seen in this case as unknown photographic filters applied randomly on the input image. This method uses a set of local transforms to restore the input images. Specifically, these transforms are sets of polynomials of degree two, whose parameters are determined through the use of a convolutional neural network. In this context, the method can be tuned with a parameter toward accuracy or speed, for matching the needs of the final user. To assess the effectiveness of the method, we processed a subset of the Places-205 dataset with 22 different photo-graphic filters. The quality of the reconstructions we obtained, measured with several objective quality measures, clearly outperformed that of the other algorithms included in the evaluation.

To address the lack of public data that is suffered in this field, it has been presented in part II a totally new method for data augmentation able to generate thousands of new samples starting from a few. We show how systems trained only on the generated synthetic data are able to outperform themselves trained only on real data. In particular, the proposed method is able to dramatically boost the performances on classes with low cardinality, that in the original formulation with real data achieved 0% of accuracy.

As an extension of the proposed pipeline, it is possible to determine a system for evaluating the degree of seriousness of the anomaly, to give a more accurate feedback to human operators. This further extension will require an in-depth study on how to assign a criticity level to an anomaly and an intensive manual data collection for extending existing datasets. This step will be very challenging since usually the criticity level is associated to a specific defective class, such as holes, and evaluated through the measurement of some characteristics of the class. In the case of anomalies, there are no assumptions on the structure and therefore there aren't measurable properties.

References

[1] (1989). *Oxford English Dictionary*. Oxford University Press.

[2] Aiger, D. and Talbot, H. (2012). The phase only transform for unsupervised surface defect detection. In *Emerging Topics In Computer Vision And Its Applications*, pages 215–232. World Scientific.

[3] Alvarez, J. and Petersson, L. (2016). Decomposeme: Simplifying convnets for end-to-end learning. *arXiv preprint arXiv:1606.05426*.

[4] An, J. and Cho, S. (2015). Variational autoencoder based anomaly detection using reconstruction probability. *Special Lecture on IE*, 2:1–18.

[5] An, X. and Pellacini, F. (2008). Appprop: all-pairs appearance-space edit propagation. In *ACM Transactions on Graphics (TOG)*, volume 27, page 40.

[6] Antoniou, A., Storkey, A., and Edwards, H. (2017). Data augmentation generative adversarial networks. *arXiv preprint arXiv:1711.04340*.

[7] Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein gan. *arXiv preprint arXiv:1701.07875*.

[8] Arthur, D. and Vassilvitskii, S. (2007). k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics.

[9] Azadi, S., Pathak, D., Ebrahimi, S., and Darrell, T. (2018). Compositional gan: Learning conditional image composition. *arXiv preprint arXiv:1807.07560*.

[10] Bae, S., Paris, S., and Durand, F. (2006). Two-scale tone management for photographic look. *ACM Transactions on Graphics (TOG)*, 25(3):637–645.

[11] Banavar, G. S. (2016). Cognitive computing: From breakthroughs in the lab to applications on the field. In *Big Data (Big Data), 2016 IEEE International Conference on*, pages 1–1. IEEE.

[12] Bartels, R. H., Beatty, J. C., and Barsky, B. A. (1998). *An Introduction to Splines for Use in Computer Graphics and Geometric Modelling*, chapter Hermite and cubic spline interpolation.

[13] Bianco, S., Buzzelli, M., and Schettini, R. (2018). Multiscale fully convolutional network for image saliency. *Journal of Electronic Imaging*, 27(5):051221.

[14] Bianco, S., Celona, L., Napoletano, P., and Schettini, R. (2017a). On the use of deep learning for blind image quality assessment. *Journal of Signal, Image and Video Processing*, -(-).

[15] Bianco, S., Cusano, C., Piccoli, F., and Schettini, R. (2017b). Artistic photo filter removal using convolutional neural networks. *Journal of Electronic Imaging*, 27(1):011004.

[16] Bianco, S., Cusano, C., and Schettini, R. (2015). Color constancy using cnns. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 81–89.

[17] Bianco, S., Cusano, C., and Schettini, R. (2017c). Single and multiple illuminant estimation using convolutional neural networks. *IEEE Transactions on Image Processing*, 26(9):4347–4362.

[18] Bianco, S., Gasparini, F., Schettini, R., and Vanneschi, L. (2008). Polynomial modeling and optimization for colorimetric characterization of scanners. *Journal of Electronic Imaging*, 17(4):043002.

[19] Bianco, S. and Schettini, R. (2014). Error-tolerant color rendering for digital cameras. *Journal of Mathematical Imaging and Vision*, 50(3):235–245.

[20] Boracchi, G., Carrera, D., and Wohlberg, B. (2014). Novelty detection in images by sparse representations. In *Intelligent Embedded Systems (IES), 2014 IEEE Symposium on*, pages 47–54. IEEE.

[21] Botta, A., De Donato, W., Persico, V., and Pescapé, A. (2016). Integration of cloud computing and internet of things: a survey. *Future Generation Computer Systems*, 56:684–700.

[22] Buchsbaum, G. (1980). A spatial processor model for object colour perception. *Journal of the Franklin Institute*, 310(1):1–26.

[23] Buslaev, A., Parinov, A., Khvedchenya, E., Iglovikov, V. I., and Kalinin, A. A. (2018). Albumentations: fast and flexible image augmentations. *arXiv preprint arXiv:1809.06839*.

[24] Bychkovsky, V., Paris, S., Chan, E., and Durand, F. (2011). Learning photographic global tonal adjustment with a database of input/output image pairs. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 97–104.

[25] Carrera, D., Manganini, F., Boracchi, G., and Lanzarone, E. (2017). Defect detection in sem images of nanofibrous materials. *IEEE Transactions on Industrial Informatics*, 13(2):551–561.

[26] Caruana, R. (1997). Multitask learning. *Machine learning*, 28(1):41–75.

[27] Chan, C.-h. and Pang, G. K. (2000). Fabric defect detection by fourier analysis. *IEEE transactions on Industry Applications*, 36(5):1267–1276.

[28] Cohen-Or, D., Sorkine, O., Gal, R., Leyvand, T., and Xu, Y.-Q. (2006). Color harmonization. In *ACM Transactions on Graphics (TOG)*, volume 25, pages 624–630.

[29] Commision Internationale de l'Eclairage, C. (1995). Industrial colour-difference evaluation. *CIE. Publication No.116*.

[30] Cubuk, E. D., Zoph, B., Mane, D., Vasudevan, V., and Le, Q. V. (2018). Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501*.

[31] Cusano, C., Napoletano, P., and Schettini, R. (2013). Intensity and color descriptors for texture classification. In *Image Processing: Machine Vision Applications VI*, volume 8661, page 866113. SPIE.

[32] Cusano, C., Napoletano, P., and Schettini, R. (2016a). Combining multiple features for color texture classification. *Journal of Electronic Imaging*, 25(6):1–9.

[33] Cusano, C., Napoletano, P., and Schettini, R. (2016b). Evaluating color texture descriptors under large variations of controlled lighting conditions. *Journal of the Optical Society of America A*, 33(1):17–30.

[34] de Boor, C. (1978). *A Practical Guide to Spline*, volume 27.

[35] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255. IEEE.

[36] Donchev, T. and Farkhi, E. (1998). Stability and euler approximation of one-sided lipschitz differential inclusions. *SIAM journal on control and optimization*, 36(2):780–796.

[37] Du, B. and Zhang, L. (2011). Random-selection-based anomaly detector for hyperspectral imagery. *IEEE Transactions on Geoscience and Remote Sensing*, 49(5):1578.

[38] Ehret, T., Davy, A., Morel, J.-M., and Delbracio, M. (2018). Image anomalies: a review and synthesis of detection methods. *arXiv preprint arXiv:1808.02564*.

[39] El-Hakim, S. F. and Pizzi, N. J. (1993). Multicamera vision-based approach to flexible feature measurement for inspection and reverse engineering. *Optical Engineering*, 32(9):2201–2216.

[40] Eslami, A. M. (2011). Computer digital image processing in quality inspection-reverse engineering approach. In *American Society for Engineering Education*. American Society for Engineering Education.

[41] Fairchild, M. D. (2013). *Color appearance models*. John Wiley & Sons.

[42] Gatys, L., Ecker, A. S., and Bethge, M. (2015). Texture synthesis using convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 262–270.

[43] Gharbi, M., Chen, J., Barron, J. T., Hasinoff, S. W., and Durand, F. (2017). Deep bilateral learning for real-time image enhancement. *ACM Transactions on Graphics (TOG)*, 36(4):118.

[44] Gilchrist, A. (2016). Introducing industry 4.0. In *Industry 4.0*, pages 195–215. Springer.

[45] Goldman, A. and Cohen, I. (2004). Anomaly detection based on an iterative local statistics approach. *Signal Processing*, 84(7):1225–1229.

[46] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

[47] Grosjean, B. and Moisan, L. (2009). A-contrario detectability of spots in textured backgrounds. *Journal of Mathematical Imaging and Vision*, 33(3):313.

[48] Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. (2017). Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pages 5767–5777.

[49] Guo, X., Li, Y., and Ling, H. (2017). Lime: Low-light image enhancement via illumination map estimation. *IEEE Transactions on Image Processing*, 26(2):982–993.

[50] Hawkins, S., He, H., Williams, G., and Baxter, R. (2002). Outlier detection using replicator neural networks. In *International Conference on Data Warehousing and Knowledge Discovery*, pages 170–180. Springer.

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

[52] Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507.

[53] Honda, T. and Nayar, S. K. (2001). Finding" anomalies" in an arbitrary image. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 2, pages 516–523. IEEE.

[54] Hu, Y., He, H., Xu, C., Wang, B., and Lin, S. (2018). Exposure: A white-box photo post-processing framework. *ACM Transactions on Graphics (TOG)*, 37(2):26.

[55] Huang, H., Yu, P. S., and Wang, C. (2018). An introduction to image synthesis with generative adversarial nets. *arXiv preprint arXiv:1803.04469*.

[56] Ignatov, A., Kobyshev, N., Timofte, R., Vanhoey, K., and Gool, L. V. (2017). Dslr-quality photos on mobile devices with deep convolutional networks. In *The IEEE International Conference on Computer Vision (ICCV)*.

[57] International Electrotechnical Commission, I. (1999). Iec 61966-2-1, 1999. *Multimedia systems and equipment — Colour measurements and management — Part 2-1: Colour management — Default RGB color space — sRGB'*.

[58] ISO 9000:2005 (2005). Quality management systems - fundamentals and vocabulary. Standard, International Organization for Standardization, Geneva, CH.

[59] Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. (2017a). Image-to-image translation with conditional adversarial networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5967–5976.

[60] Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. (2017b). Image-to-image translation with conditional adversarial networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[61] Julesz, B. (1981). Textons, the elements of texture perception, and their interactions. *Nature*, 290(5802):91.

[62] Kang, H. R. (1997). *Color technology for electronic imaging devices*. SPIE press.

[63] Kang, S. B., Kapoor, A., and Lischinski, D. (2010). Personalization of image enhancement. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1799–1806.

[64] Kaplinsky, R. (1985). Electronics-based automation technologies and the onset of systemofacture: Implications for third world industrialization. *World Development*, 13(3):423–439.

[65] Kaufman, L., Lischinski, D., and Werman, M. (2012). Content-aware automatic photo enhancement. In *Computer Graphics Forum*, volume 31, pages 2528–2540.

[66] Khaitan, S. K. and McCalley, J. D. (2015). Design techniques and applications of cyberphysical systems: A survey. *IEEE Systems Journal*, 9(2):350–365.

[67] Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In *International Conference on Learning Representation*.

[68] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

[69] Kumar, A. (2008). Computer-vision-based fabric defect detection: A survey. *IEEE transactions on industrial electronics*, 55(1):348–363.

[70] Kumar, A. and Pang, G. K. (2002). Defect detection in textured materials using gabor filters. *IEEE Transactions on industry applications*, 38(2):425–440.

[71] Land, E. H. and McCann, J. J. (1971). Lightness and retinex theory. *Journal of the Optical Society of America A*, 61(1):1–11.

[72] Lasi, H., Fettke, P., Kemper, H.-G., Feld, T., and Hoffmann, M. (2014). Industry 4.0. *Business & Information Systems Engineering*, 6(4):239–242.

[73] LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436.

[74] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

[75] Ledig, C., Theis, L., Huszár, F., Caballero, J., Cunningham, A., Acosta, A., Aitken, A. P., Tejani, A., Totz, J., Wang, Z., et al. (2017). Photo-realistic single image super-resolution using a generative adversarial network. In *CVPR*, volume 2, page 4.

[76] Levina, E. and Bickel, P. (2001). The earth mover? s distance is the mallows distance: Some insights from statistics. In *null*, page 251. IEEE.

[77] Lim, S. K., Loo, Y., Tran, N.-T., Cheung, N.-M., Roig, G., and Elovici, Y. (2018). Doping: Generative data augmentation for unsupervised anomaly detection with gan. *arXiv preprint arXiv:1808.07632*.

[78] Lischinski, D., Farbman, Z., Uyttendaele, M., and Szeliski, R. (2006). Interactive local adjustment of tonal values. In *ACM Transactions on Graphics (TOG)*, volume 25, pages 646–653.

[79] Long, J., Shelhamer, E., and Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440.

[80] Maaten, L. v. d. and Hinton, G. (2008). Visualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605.

[81] Margolin, R., Tal, A., and Zelnik-Manor, L. (2013). What makes a patch distinct? In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1139–1146.

[82] Mazzini, D. (2018). Guided upsampling network for real-time semantic segmentation. In *British Machine Vision Conference*.

[83] Mirza, M. and Osindero, S. (2014). Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*.

[84] Napoletano, P. (2017). Hand-crafted vs learned descriptors for color texture classification. In *International Workshop on Computational Color Imaging*, pages 259–271. Springer.

[85] Napoletano, P. (2018). Visual descriptors for content-based retrieval of remote-sensing images. *International Journal of Remote Sensing*, 39(5):1–34.

[86] Napoletano, P., Piccoli, F., and Schettini, R. (2018). Anomaly detection in nanofibrous materials by cnn-based self-similarity. *Sensors*, 18(1):209.

[87] Nash, J. F. et al. (1950). Equilibrium points in n-person games. *Proceedings of the national academy of sciences*, 36(1):48–49.

[88] Oesterreich, T. D. and Teuteberg, F. (2016). Understanding the implications of digitisation and automation in the context of industry 4.0: A triangulation approach and elements of a research agenda for the construction industry. *Computers in Industry*, 83:121–139.

[89] Paszke, A., Chaurasia, A., Kim, S., and Culurciello, E. (2016). Enet: A deep neural network architecture for real-time semantic segmentation. *arXiv preprint arXiv:1606.02147*.

[90] Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. (2017). Automatic differentiation in pytorch. In *Autodiff Workshop — NIPS 2017*.

[91] Perng, D.-B., Chen, S.-H., and Chang, Y.-S. (2010). A novel internal thread defect auto-inspection system. *The International Journal of Advanced Manufacturing Technology*, 47(5-8):731–743.

[92] Pimentel, M. A., Clifton, D. A., Clifton, L., and Tarassenko, L. (2014). A review of novelty detection. *Signal Processing*, 99:215–249.

[93] Radford, A., Metz, L., and Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*.

[94] Razavian, A., Azizpour, H., Sullivan, J., and Carlsson, S. (2014). CNN features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 806–813.

[95] Romera, E., Alvarez, J. M., Bergasa, L. M., and Arroyo, R. (2018). Erfnet: Efficient residual factorized convnet for real-time semantic segmentation. *IEEE Transactions on Intelligent Transportation Systems*, 19(1):263–272.

[96] Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 234–241.

[97] Rüschendorf, L. (2001). Wasserstein metric. *Hazewinkel, Michiel, Encyclopaedia of Mathematics, Springer*.

[98] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. (2015). Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252.

[99] Schlegl, T., Seeböck, P., Waldstein, S. M., Schmidt-Erfurth, U., and Langs, G. (2017). Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. In *International Conference on Information Processing in Medical Imaging*, pages 146–157. Springer.

[100] Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117.

[101] Schwartz, E., Karlinsky, L., Shtok, J., Harary, S., Marder, M., Feris, R., Kumar, A., Giryes, R., and Bronstein, A. M. (2018). Delta-encoder: an effective sample synthesis method for few-shot object recognition. *arXiv preprint arXiv:1806.04734*.

[102] Shi, W., Caballero, J., Huszár, F., Totz, J., Aitken, A. P., Bishop, R., Rueckert, D., and Wang, Z. (2016). Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1874–1883.

[103] Simonyan, K. and Zisserman, A. (2014a). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

[104] Simonyan, K. and Zisserman, A. (2014b). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

[105] Sironi, A., Tekin, B., Rigamonti, R., Lepetit, V., and Fua, P. (2015). Learning separable filters. *IEEE transactions on pattern analysis and machine intelligence*, 37(1):94–106.

[106] Stock, T. and Seliger, G. (2016). Opportunities of sustainable manufacturing in industry 4.0. *Procedia Cirp*, 40:536–541.

[107] Szegedy, C., Ioffe, S., Vanhoucke, V., and Alemi, A. A. (2017). Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, volume 4, page 12.

[108] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9.

[109] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826.

[110] Tarassenko, L., Hayton, P., Cerneaz, N., and Brady, M. (1995). Novelty detection for the identification of masses in mammograms.

[111] Tout, K., Cogranne, R., and Retraint, F. (2016). Fully automatic detection of anomalies on wheels surface using an adaptive accurate model and hypothesis testing theory. In *Signal Processing Conference (EUSIPCO), 2016 24th European*, pages 508–512. IEEE.

[112] Tsai, D.-M. and Hsieh, C.-Y. (1999). Automated surface inspection for directional textures. *Image and vision computing*, 18(1):49–62.

[113] Valdeza, A. C., Braunera, P., Schaara, A. K., Holzingerb, A., and Zieflea, M. (2015). Reducing complexity with simplicity-usability methods for industry 4.0. In *Proceedings 19th triennial congress of the IEA*, volume 9, page 14.

[114] Vedaldi, A. and Lenc, K. (2014). Matconvnet – convolutional neural networks for matlab. *CoRR*, abs/1412.4564.

[115] Wang, D. C., Vagnucci, A. H., and Li, C. (1983). Digital image enhancement: a survey. *Computer Vision, Graphics, and Image Processing*, 24(3):363–381.

[116] Wang, Z., Bovik, A. C., Sheikh, H. R., and Simoncelli, E. P. (2004). Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612.

[117] Wheeler, D. A., Brykczynski, B., and Meeson Jr, R. N. (1996). *Software Inspection: An Industry Best Practice for Defect Detection and Removal*. IEEE Computer Society Press.

[118] Wold, S., Esbensen, K., and Geladi, P. (1987). Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52.

[119] Wollschlaeger, M., Sauter, T., and Jasperneite, J. (2017). The future of industrial communication: Automation networks in the era of the internet of things and industry 4.0. *IEEE Industrial Electronics Magazine*, 11(1):17–27.

[120] Xie, X. and Mirmehdi, M. (2005). Texture exemplars for defect detection on random textures. In *International Conference on Pattern Recognition and Image Analysis*, pages 404–413. Springer.

[121] Yan, Z., Zhang, H., Wang, B., Paris, S., and Yu, Y. (2016). Automatic photo adjustment using deep neural networks. *ACM Transactions on Graphics (TOG)*, 35(2):11.

[122] Yang, J. and Yang, M.-H. (2017). Top-down visual saliency via joint crf and dictionary learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(3):576–588.

[123] Zhang, X. and Wandell, B. A. (1997). A spatial extension of CIELAB for digital color-image reproduction. *Journal of the Society for Information Display*, 5(1):61–63.

[124] Zhao, H., Gallo, O., Frosio, I., and Kautz, J. (2017). Loss functions for image restoration with neural networks. *IEEE Transactions on Computational Imaging*, 3(1):47–57.

[125] Zhou, B., Lapedriza, A., Xiao, J., Torralba, A., and Oliva, A. (2014). Learning deep features for scene recognition using places database. In *Advances in Neural Information Processing Systems*, pages 487–495.

[126] Zhou, T., Tulsiani, S., Sun, W., Malik, J., and Efros, A. A. (2016). View synthesis by appearance flow. In *European conference on computer vision*, pages 286–301. Springer.

[127] Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A. (2017a). Unpaired image-to-image translation using cycle-consistent adversarial networks. In *IEEE International Conference on Computer Vision (ICCV)*.

[128] Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A. (2017b). Unpaired image-to-image translation using cycle-consistent adversarial networks. In *The IEEE International Conference on Computer Vision (ICCV)*.

Appendix A

Datasets

A.1 Artistic Foto Filter Dataset

We have built this dataset with the intent of making available to the researchers' community a reference dataset upon which is possible to banchmark methods for detecting spatially distributed anomalies and also for the task of artistic photo filter removal. This dataset is built starting from the Places-205 data set[125]. Places-205 has been designed to represent places and scenes found in the real world. It includes over one million images labeled with 205 different categories (each category is represented by at least 5000 images). See section A.3 for more details on the Places205 dataset. For creating this dataset, we randomly sampled 20 000 images. After that, we processed them with the 22 filters (explained in subsection A.1.1) to form a dataset of 460 000 filtered images (including the original ones). Figure A.1 shows a random sample draw from this dataset. The Images were randomly divided into training, validation and test sets with ratios 75%, 5%, and 20% having care to place all the filtered variants of the same image in the same set.

A.1.1 Photographic filters

Many photo sharing services, such as Instagram®, give to their users the option to apply photographic filters to their own pictures. These filters consist of a pipeline of photo editing operations that are applied in a completely automatic way. Editing operations that are often used in photographic filters include: global transformations of the color distribution by using "color levels", "color curves", or by changing the pixels' hue, saturation and lightness; global adjustment of the image brightness and contrast; introduction of blur or noise; introduction of a "vignette" effect (i.e. darkening of the

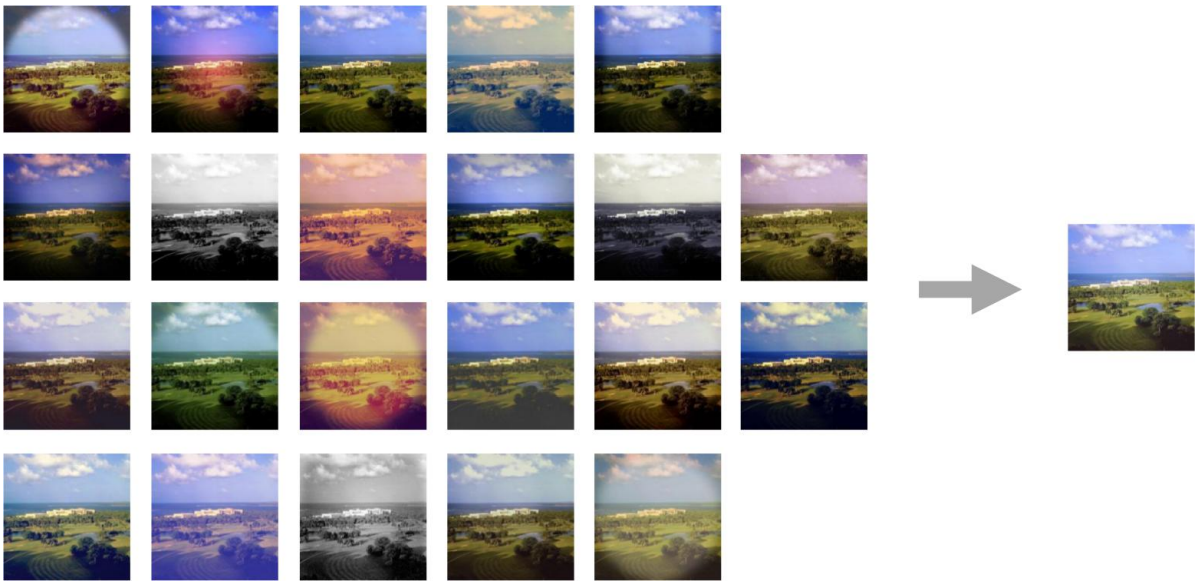


Fig. A.1 A sample from the Artistic Photo Filter Dataset. On the left, the 22 versions of the filtered image with instagram-like filters, on the right the corresponding image without filters (the ground truth).

border of the image); spatially varying modification of color with the use of a gradient; conversion to Black & White; introduction of a "flare" effect in the central part of the image.

In this dataset we considered 22 photographic filters defined by photo editing enthusiasts to match those made available by the Instagram® photo sharing service. In particular, we considered the filters named *1977*, *Amaro*, *Apollo*, *Brannan*, *Earlybird*, *Gotham*, *Hefe*, *Hudson*, *Inkwell*, *Lofi*, *Lord Kelvin*, *Mayfair*, *Nashville*, *Poprocket*, *Rise*, *Sierra*, *Sutro*, *Toaster*, *Valencia*, *Walden*, *Willow*, and *Xpro-II*. As a special case of filter, we also included the original images without any further processing. Table A.1 summarizes the filters in terms of their editing operations, while Figure A.2 reports, for each one, a brief description taken from the Instagram® website.

A.2 Nanofibers Dataset

This dataset is composed by 45 images of nanofibrous materials acquired with a SEM (Scanning Electron Microscope). The external appearance of this material can be seen as a non-periodic continuous texture with intertwined filamentous elements that look like white wires, some examples without and with anomalies are in Fig. A.3. The dataset is composed by two disjoint subsets: a set of 5 images without anomalies, that we call

Table A.1 Summary of the basic image processing operations used in the 23 photographic filters considered.

Filter name	Color levels	Color curves	Brightness/Contrast	Blur/Noise	Hue/Sat/Lightness	Vignette	Color layer	Gradient	Black & White	Flare
Original	·	·	·	·	·	·	·	·	·	·
1977	·	✓	·	·	·	·	·	·	·	·
Amaro	·	✓	·	·	·	✓	·	·	·	·
Apollo	·	·	·	·	·	✓	✓	·	·	·
Brannan	·	✓	✓	·	✓	·	·	·	·	·
Earlybird	·	✓	✓	·	✓	✓	✓	·	·	·
Gotham	·	✓	·	✓	·	·	·	·	✓	·
Hefe	·	·	✓	·	✓	✓	·	·	·	·
Hudson	·	✓	·	·	·	·	✓	·	·	·
Inkwell	·	✓	✓	·	·	·	·	·	✓	·
Lofi	·	✓	✓	·	·	✓	·	✓	·	·
Lord Kelvin	·	✓	·	·	·	·	·	·	·	·
Mayfair	✓	✓	·	✓	✓	·	·	·	·	·
Nashville	✓	✓	✓	·	·	·	·	·	·	·
Poprocket	·	·	·	·	·	·	·	✓	·	·
Rise	·	·	·	✓	✓	✓	✓	✓	·	·
Sierra	·	✓	·	·	·	✓	·	✓	·	·
Sutro	·	✓	✓	·	✓	✓	✓	·	·	·
Toaster	·	✓	·	·	·	✓	✓	✓	·	·
Valencia	✓	✓	·	·	·	·	·	·	·	·
Walden	✓	✓	·	·	·	·	·	✓	·	·
Willow	·	·	·	✓	·	✓	·	✓	✓	✓
X-pro II	·	✓	·	·	·	·	·	·	·	·

Original: image taken from the Places data base without any additional processing.

1977: the increased exposure with a red tint gives the photograph a rosy, brighter, faded look.

Amaro: adds light to an image, with the focus on the centre.

Apollo: lightly bleached, cyan-greenish color, some dusty texture.

Brannan: increases contrast and exposure and adds a metallic tint.

Earlybird: gives photographs an older look with a sepia tint and warm temperature.

Gotham: produce a black and white high contrast image, with bluish undertones.

Hefe: hight contrast and saturation, with a similar effect to Lo-Fi but not quite as dramatic.

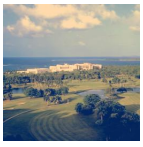
Hudson: creates an "icy" illusion with heightened shadows, cool tint and dodged center.

Inkwell: direct shift to black and white — no extra editing.

Lord Kelvin: increases saturation and temperature to give it a radiant "glow".

Lo-fi: enriches color and adds strong shadows through the use of saturation and "warming" the temperature.

Mayfair: applies a warm pink tone, subtle vignetting to brighten the photograph center and a thin black border.



Nashville: warms the temperature, lowers contrast and increases exposure to give a light "pink" tint — making it feel "nostalgic".

Poprocket: adds a creamy vintage and retro color effect.

Rise: adds a "glow" to the image, with softer lighting of the subject.

Sierra: gives a faded, softer look.

Sutro: burns photo edges, increases highlights and shadows dramatically with a focus on purple and brown colors.

Toaster: ages the image by "burning" the centre and adds a dramatic vignette.

Valencia: fades the image by increasing exposure and warming the colors, to give it an antique feel.

Walden: increases exposure and adds a yellow tint.

Willow: a monochromatic filter with subtle purple tones and a translucent white border.

X-Pro II: increases color vibrance with a golden tint, high contrast and slight vignette added to the edges.

Fig. A.2 Examples of the photographic filters considered in this work. The text has been taken from the Instagram® website.

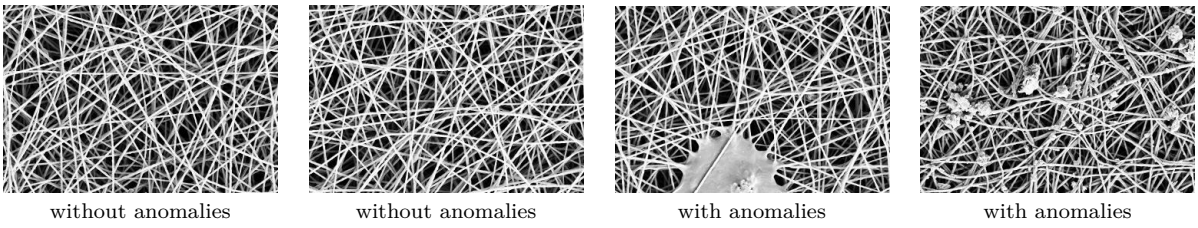


Fig. A.3 Examples of SEM images of nanofibrous materials with and without anomalies.

Fig. A.4 (a) The input image $\mathbf{I}$. (b) The binary mask of anomalies $\Omega_{\mathbf{I}}$. White pixels represent anomalies. (c) Estimated mask of anomalies $\widetilde{\Omega_{\mathbf{I}}}$. White pixels represent anomalies. (d) Difference between $\Omega_{\mathbf{I}}$ and $\widetilde{\Omega_{\mathbf{I}}}$ overlaid on the test image. Green pixels represent *true positives*, red pixels represent *false positives*, blue pixels represent *false negatives*, no color pixels represent *true negatives*.

"normal", and a set 40 of images with defects that we call anomalies. All the defects have been manually annotated. The dataset and the defect annotations are publicly available at http://web.mi.imati.cnr.it/ettore/NanoTwice. Each image $\mathbf{I}$ is gray-scale and of size 700×1024. The annotations associated to each anomalous image is a map $\Omega_{\mathbf{I}} \in \{0, 1\}$. Figure A.4 (a) and (b) shows an anomalous image along with its defect annotation. The set of normal images $\mathcal{I}^{normal}$ is divided in two subsets $\mathcal{I}^{train}$, $\mathcal{I}^{val}$, which are respectively used to create the dictionary and to assess through visual self similarity the value of the threshold th used during the test time to convert the aggregated distance map $\Theta_{\mathbf{I}}$ in the corresponding anomaly map $\widetilde{\Omega_{\mathbf{I}}}$. $\mathcal{I}^{train}$ and $\mathcal{I}^{val}$ have respectively a cardinality of 4 and 1. The set of test images $\mathcal{I}^{test}$ contains 40 images with anomalies. For sake of comparison with the state of the art, we use a subset made of 35 images obtained by removing images that are listed in $\mathcal{I}^{test}$ at the following positions: 8, 15, 27, 31, 35. These images containing anomalies are required by the state of the art methods to validate the parameters. Our method dos not require to use images containing anomalies at any time a part from testing.

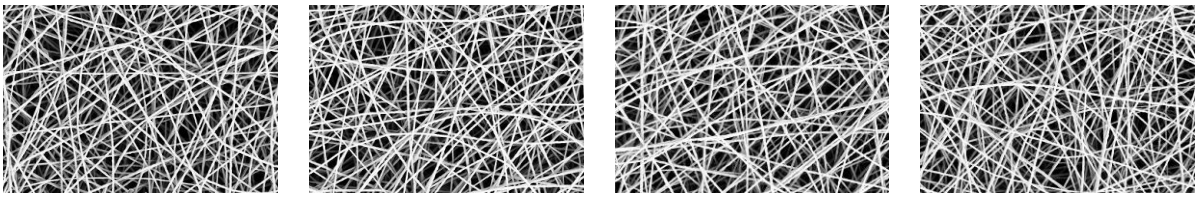


Fig. A.5 This is the trainset. Is composed by 4 images of normal samples, i.e. samples that do not present anomalies.

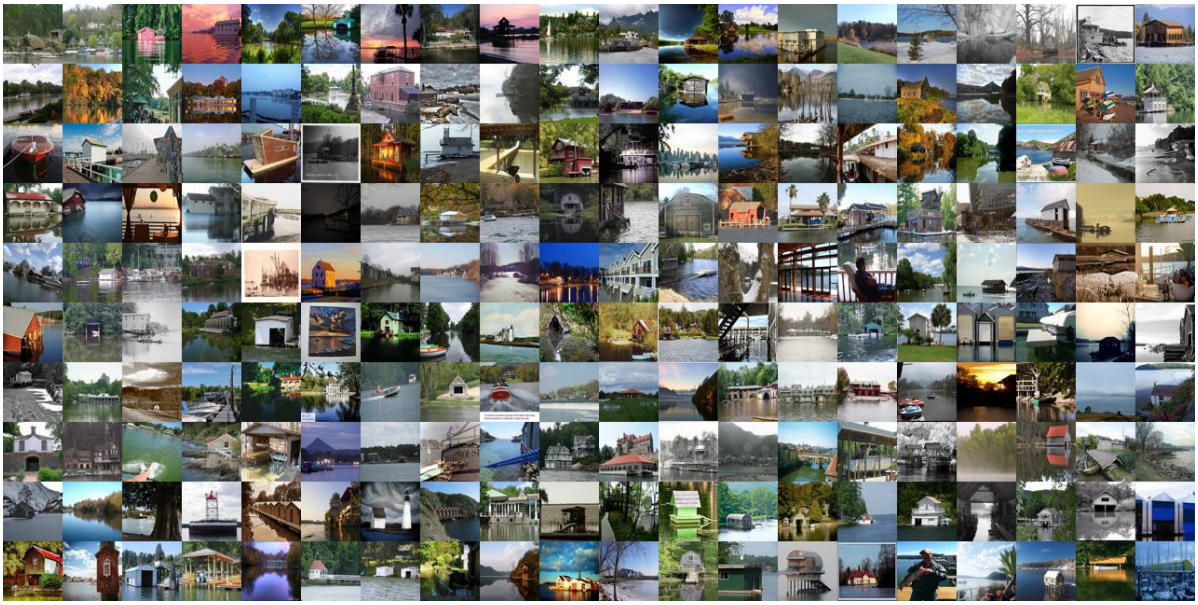


Fig. A.6 Places205 dataset samples

A.3 Places205 Dataset

The places205 dataset [125] is a scene-centric database, with 205 scene categories and 2.5 millions of images with a category label. Figure A.6 shows sample images of this dataset.

A.4 FiveK Dataset

It's the MIT-Adobe FiveK dataset [24] (in short FiveK). It contains 5000 samples. Each sample is composed by a raw image in the *DNG* format and by the same image enhanced by five experts in *16-bit TIFF* format. The dataset is also distributed as a single Adobe Lightroom® catalog, which makes the download easier and gives the possibility to explore the history of the enhancement steps performed by the experts. The dataset has been used in many works but unfortunately each work has exported the images from the catalog in a different, and often non-documented, way. In order to limit the diffusion of

Fig. A.7 A sample from the FiveK dataset. For each raw image the dataset includes five different versions retouched by five expert photographers.

multiple, different, copies of the dataset, to convert the *DNG* files in a python-readable format and to allow the comparison of our method with the state-of-the-art, we converted the images using the procedure described in [54], which is publicly documented and available online: starting from the catalog, in the collection list we select the collection *Inputs/Input with Daylight WhiteBalance minus 1.5*; the raw images are exported as *16-bit TIFF* in the *ProPhotoRGB* color space and then converted into the *sRGB* color space as *8-bit PNG*. This last conversion is required to make it possible to correctly interpret the images with common image processing libraries. The experts' images are exported as *8-bit PNG* in the *sRGB* color space. From the same work by Hu *et al* we also take the subdivision into a training set of 4000 images and a test set of 1000 images.

A sample image from the FiveK dataset and the corresponding versions retouched by the five expert photographers are reported in Figure A.7, where it can be noticed the different style of the five experts.

Appendix B

CIELab conversion

We report here the formulae used for the conversion from *sRGB* to *CIELab* color space. It is assumed that the input *sRGB* values are normalized in the $[0, 1]$ range.

First, the companded RGB channels (denoted with upper case (R, G, B), or generically as V to indicate each channel indifferently) are made linear with respect to energy (denoted with lower case (r, g, b), or generically as v to indicate each channel indifferently):

$$v = \begin{cases} V/12.92 & \text{if } V \leq 0.04045, \\ ((V + 0.055)/1.055)^{2.4} & \text{otherwise.} \end{cases} \tag{B.1}$$

The next step is the conversion of obtained linear RGB values to XYZ color space:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.4124564 & 0.3575761 & 0.1804375 \\ 0.2126729 & 0.7151522 & 0.0721750 \\ 0.0193339 & 0.1191920 & 0.9503041 \end{bmatrix} \cdot \begin{bmatrix} r \\ g \\ b \end{bmatrix}. \tag{B.2}$$

Then using as the reference white the *CIE D65* standard illuminant ($[X_r, Y_r, Z_r] = [0.95047, 1.0000, 1.08883]$), the white-normalized XYZ values are obtained:

$$[x_r, y_r, z_r] = \left[\frac{X}{X_r}, \frac{Y}{Y_r}, \frac{Z}{Z_r}\right]. \tag{B.3}$$

These values are then non-linearly transformed:

$$f_c = \begin{cases} c^{1/3} & \text{if } c > \epsilon, \\ \frac{\kappa c + 16}{116} & \text{otherwise,} \end{cases} \tag{B.4}$$

where $c = (x_r, y_r, z_r)$, $\epsilon = 0.008856$ and $\kappa = 903.3$. Finally, the last step converts the values to *CIELab*:

$$L = 116 f_y - 16, \tag{B.5}$$

$$a = 500(f_x - f_y), \tag{B.6}$$

$$b = 200(f_y - f_z). \tag{B.7}$$

Appendix C

Spline Interpolation

A spline is a piecewise function composed by $N-1$ polynomial sub-functions of degree three, expressed in the canonical form:

$$y_i(x) = a_i (x - x_i)^3 + b_i (x - x_i)^2 + c_i (x - x_i) + d_i \tag{C.1}$$

where N is the number of nodes and $i \in [0, N)$.

Once the nodes of the spline are determined we need to find the coefficients a_i, b_i, c_i and d_i of each polynomial. Since there are N nodes and consequently $(N-1)$ polynomials, there are $4\,(N-1)$ variables and $(N-1)$ equations. Let:

$$s_i'(x) = 3a_i (x - x_i)^2 + 2b_i (x - x_i) + c_i \tag{C.2}$$

be the first derivative and

$$s_i''(x) = 6a_i (x - x_i) + 2b_i \tag{C.3}$$

be the second derivative of each $i-th$ polynomial described in eq. C.1 with respect to x. By introducing a constraint on continuity:

$$s_i (x_i) = s_{i-1} (x_i), \tag{C.4}$$

a constraint on differentiability:

$$s_i' (x_i) = s_{i-1}' (x_i) \tag{C.5}$$

and finally a constraint on smoothness:

$$s_i'' (x_i) = s_{i-1}'' (x_i) \tag{C.6}$$

between adjacent polynomials it's possible to make this under-determined system solvable. Furthermore, the constraints on continuity and differentiability make the splines suitable for gradient-descent based methods such as the convolutional neural networks.

From equation C.1 and from the constraint on continuity C.1 we derive:

$$s_i(x_i) = d_i \tag{C.7}$$

and

$$s_{i-1}(x_i) = a_{i-1}(x_i - x_{i-1})^3 + b_{i-1}(x_i - x_{i-1})^2 + c_{i-1}(x_i - x_{i-1}) + d_{i-1} \tag{C.8}$$

so

$$d_i = a_{i-1}(x_i - x_{i-1})^3 + b_{i-1}(x_i - x_{i-1})^2 + c_{i-1}(x_i - x_{i-1}) + d_{i-1}\,. \tag{C.9}$$

for $i \in [2, N]$. Letting $h = x_i - x_{i-1}$, in equation C.8, we obtain:

$$d_i = a_{i-1}h^3 + b_{i-1}h^2 + c_{i-1}h + d_{i-1} \tag{C.10}$$

From the first derivative in equation C.2 and the differentiability constraint (eq. C.5), we have that:

$$s_i'(x_i) = c_i \tag{C.11}$$

and

$$s_{i-1}' = 3a_{i-1}(x_i - x_{i-1})^2 + 2b_{i-1}(x_i - x_{i-1}) + c_{i-1} \tag{C.12}$$

Again letting $h = x_i - x_{i-1}$, we obtain:

$$c_i = 3a_{i-1}h^2 + 2b_{i-1}h + c_{i-1} \tag{C.13}$$

for $i \in [2, N-1]$. Finally, from the second derivative (eq. C.3) and from the smoothness constraint in eq. C.6, we gather:

$$s_i''(x_i) = 2b_i \tag{C.14}$$

and

$$\begin{aligned} s_i''(x_{i+1}) &= s_{i+1}''(x_{i+1}) = \\ &= 6a_i(x_{i+1} - x_i) + 2b_i \end{aligned} \tag{C.15}$$

and letting again $h = x_i - x_{i-1}$, from equations C.14 and C.15 we get:

$$s_{i+1}'' = 6a_i(x_{i+1} - x_i) + 2b_i \tag{C.16}$$

$$2b_{i+1} = 6a_i h + 2b_i \tag{C.17}$$

By doing a variable change, it's possible to simplify the equations. Specifically, by introducing:

$$M_i = s''(x_i) \tag{C.18}$$

it's possible to rewrite all the parameters. Each b_i can be represented by:

$$\begin{aligned} s''(x_i) &= 2b_i \\ M_i &= 2b_i \\ b_i &= \frac{M_i}{2} \end{aligned} \tag{C.19}$$

Similarily, using equation C.17, a_i can be rewritten as:

$$\begin{aligned} 2b_{i+1} &= 6a_i h + 2b_i \\ 6a_i h &= 2b_{i+1} - 2b_i \\ a_i &= \frac{2b_{i+1} - 2b_i}{6h} \\ a_i &= \frac{2\left(\frac{M_{i+1}}{2}\right) - 2\left(\frac{M_i}{2}\right)}{6h} \\ a_i &= \frac{M_{i+1} - M_i}{6h} \end{aligned} \tag{C.20}$$

In the same way, c_i can be re-written as:

$$\begin{aligned} d_{i+1} &= a_i h^3 + b_i h^2 + c_i h + d_i \\ c_i h &= -a_i h^3 - b_i h^2 - d_i - d_{i+1} \\ c_i &= \frac{-a_i h^3 - b_i h^2 - d_i - d_{i+1}}{h} \\ c_i &= \frac{-a_i h^3 - b_i h^2}{h} + \frac{-d_i - d_{i+1}}{h} \\ c_i &= \left(-a_i h^2 - b_i h\right) - \frac{d_i + d_{i+1}}{h} \\ c_i &= -\left(\frac{Mi + 1 - M_i}{6h} h^2 + \frac{M_i}{2} h\right) - \frac{y_i - y_{i+1}}{h} \\ c_i &= \frac{y_{i+1} - y_i}{h} - \left(\frac{M_{i+1} - M_i}{6} h + \frac{3M_i}{6} h\right) \\ c_i &= \frac{y_{i+1} - y_i}{h} - \frac{M_{i+1} - M_i + 3M_i}{6} h \\ c_i &= \frac{y_{i+1} - y_i}{h} - \frac{M_{i+1} + 2M_i}{6} h \end{aligned} \tag{C.21}$$

Now that we computed all the equations we can rewrite the four parameters of each polynomial composing the spline as:

$$\begin{aligned} a_i &= \frac{M_{i+1} - M_i}{6h} \\ b_i &= \frac{M_i}{2} \\ c_i &= \frac{y_{i+1} - y_i}{h} - \left(\frac{M_{i+1} + 2M_i}{6}\right) h \\ d_i &= y_i \end{aligned} \tag{C.22}$$

To make the system easier to handle we need to rewrite it in a matricial form, starting from equation C.13 and sobstituting the four parameters expressed in function of M_i (equation C.22) we can write:

$$\begin{aligned} 3a_i h^2 + 2b_i h + c_i &= c_{i+1} \\ 3\left(\frac{M_{i+1} - M_i}{6h}\right) h^2 + 2\left(\frac{M_i}{2}\right) + \frac{y_{i+1} - y_i}{h} - \left(\frac{M_{i+1} + 2M_i}{6}\right) h &= \frac{y_{i+2} - y_{i+1}}{h} - \left(\frac{M_{i+2} + 2M_{i+1}}{6}\right) h \\ 3\left(\frac{M_{i+1} - M_i}{6h}\right) h^2 + 2\left(\frac{M_i}{2}\right) h - \left(\frac{M_{i+1} + 2M_i}{6}\right) h + \left(\frac{M_{i+2} + 2M_{i+1}}{6}\right) h &= -\frac{y_{i+1} - y_i}{h} + \frac{y_{i+2} - y_{i+1}}{h} \\ h\left[\frac{3M_{i+1} - 3M_i}{6} + \frac{6M_i}{6} - \left(\frac{M_{i+1} + 2M_i}{6}\right) + \left(\frac{M_{i+2} + 2M_{i+1}}{6}\right)\right] &= \frac{y_i - 2y_{i+1} + y_{i+2}}{h} \\ \frac{h}{6}\left(M_i + 4M_{i+1} + M_{i+2}\right) &= \frac{y_i - 2y_{i+1} + y_{i+2}}{h} \\ M_i + 4M_{i+1} + M_{i+2} &= \frac{6}{h^2}\left(y_i - 2y_{i+1} + y_{i+2}\right) \end{aligned} \tag{C.23}$$

for $i \in [1, N)$.

Finally, it's possible to convert it in matricial form as follows:

$$\begin{bmatrix} 1 & 4 & 1 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 1 & 4 & 1 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 1 & 4 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \\ 0 & 0 & 0 & 0 & \cdots & 4 & 1 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 1 & 4 & 1 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 1 & 4 \end{bmatrix} \begin{bmatrix} M_1 \\ M_2 \\ M_3 \\ M_4 \\ \vdots \\ M_{n-3} \\ M_{n-2} \\ M_{n-1} \\ M_n \end{bmatrix} = \\ = \frac{6}{h^2} \begin{bmatrix} y_1 - 2y_2 + y_3 \\ y_2 - 2y_3 + y_4 \\ y_3 - 2y_4 + y_5 \\ \vdots \\ y_{n-4} - 2y_{n-3} + y_{n-2} \\ y_{n-3} - 2y_{n-2} + y_{n-1} \\ y_{n-2} - 2y_{n-1} + y_n \end{bmatrix} \tag{C.24}$$

Note that the system still be under-determined. To make it solvable, we adopt a particular type of spline called *natural spline*. In this type of splines, we constraint the second derivative to be null at the endpoints:

$$M_1 = M_n = 0 \tag{C.25}$$

Equation C.24 becomes therefore:

$$\begin{bmatrix} 4 & 1 & 0 & \cdots & 0 & 0 & 0 \\ 1 & 4 & 1 & \cdots & 0 & 0 & 0 \\ 0 & 1 & 4 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 4 & 1 & 0 \\ 0 & 0 & 0 & \cdots & 1 & 4 & 1 \\ 0 & 0 & 0 & \cdots & 0 & 1 & 4 \end{bmatrix} \begin{bmatrix} M_2 \\ M_3 \\ M_4 \\ \vdots \\ M_{n-3} \\ M_{n-2} \\ M_{n-1} \end{bmatrix} = \frac{6}{h^2} \begin{bmatrix} y_1 - 2y_2 + y_3 \\ y_2 - 2y_3 + y_4 \\ y_3 - 2y_4 + y_5 \\ \vdots \\ y_{n-4} - 2y_{n-3} + y_{n-2} \\ y_{n-3} - 2y_{n-2} + y_{n-1} \\ y_{n-2} - 2y_{n-1} + y_n \end{bmatrix} \tag{C.26}$$

This equation is equivalent to $KM = \frac{6}{h^2}Y$. The solution of the overall system is therefore:

$$M = \frac{6}{h^2}K^{-1}Y \tag{C.27}$$