



PH.D. SCHOOL

UNIVERSITY OF MILANO-BICOCCA

DEPARTMENT OF INFORMATICS, SYSTEMS AND COMMUNICATION
PHD PROGRAM IN COMPUTER SCIENCE - XXX CYCLE

Local Detectors and Descriptors for Object and Scene Recognition

Ph.D. Dissertation of: Davide Mazzini

Supervisor: Prof. Raimondo Schettini

Co-Supervisor: Dr. Simone Bianco

Tutor: Prof. Andrea Maurino

Ph.D. Coordinator: Prof.ssa Stefania Bandini

ACADEMIC YEAR 2016-2017

Acknowledgements

I am very grateful to my supervisor Prof. Raimondo Schettini for his precious help, comments and corrections. For putting together a wonderful team and believe on each one of us.

Further I would like to acknowledge my co-supervisor Dr. Simone Bianco for the fruitful discussions and patience he shown during these years.

I would like to thank my colleagues of the IVL laboratry for being a constant help during the everyday lab life. For having helped my journey trough a very friendly, and for this reason, very productive and stimulating environment.

I would like also to thank my parents for their support and for being living examples of determination and humble hard working. Last, but certainly not least, I want to thank my girlfriend, and future wife, Emanuela for her support, patience and love that accompanied me throughout these years and still does.

Abstract

The aim of this thesis is to study two main categories of algorithms for object detection and their use in particular applications. The first category that is investigated concerns Keypoint-based approaches. Several comparative experiments are performed within the standard testing pipeline of the MPEG CDVS Test Model, and an extended pipeline which make use of color information is proposed. The second category of object detectors that is investigated is based on Convolutional Neural Networks. Two applications of Convolutional Neural Networks for object recognition are in particular addressed. The first concerns logo recognition. Two classification pipelines are designed and tested on a real-world dataset of images collected from Flickr. The first architecture makes use of a pre-trained network as feature extractor and it achieves comparable results keypoint based approaches. The second architecture makes use of a tiny end-to-end trained Neural Network that outperformed state-of-the-art keypoint based methods. The other application addressed is Painting Categorization. It consists in associating the author, assigning a painting to the school or art movement it belongs to, and categorizing the genre of the painting, e.g. landscape, portrait, illustration etc. To tackle this problem, a novel multibranch and multitask Neural Network structure is proposed which benefits from the joint use of keypoint-based approaches and neural features. In both applications the use of data augmentation techniques to enlarge the training set is also investigated. In particular for paintings, a neural style transfer algorithm is exploited for generating synthetic paintings to be used in training.

Table of contents

List of figures	xi
List of tables	xv
1 Introduction	1
1.1 Thesis overview	3
2 Common Local Image Transformations	5
2.1 Light change	5
2.1.1 Reflectance model	6
2.1.2 Von Kries Model	6
2.1.3 Five lighting transformations	7
2.2 Affine transformations	10
2.3 Blur, Noise and JPEG compression	12
3 Keypoint-based Approaches	15
3.1 Interest Point Detectors	15
3.1.1 Desirable properties of corner detectors	15
3.1.2 Harris detector	16
3.1.3 Scale and affine invariant harris detector	20
3.1.4 SIFT Detector	21
3.1.5 SURF detector	24
3.1.6 FAST inspired detectors	24
3.1.7 Kaze	26
3.2 Keypoint-based Local Descriptors	28
3.2.1 SIFT	29
3.2.2 SURF	30
3.2.3 FREAK	30
3.2.4 Color descriptors	32

Table of contents

3.3	Experimental Setup and Datasets	36
3.3.1	The MPEG CDVS Test Model Framework	36
3.3.2	Pairwise Matching Experiment	37
3.3.3	Datasets	39
3.3.4	SuperMarket Milan Dataset	41
3.4	Quantitative Evaluation	42
3.4.1	Algorithms considered for comparison	42
3.4.2	CDVS Evaluation on SuperMarket Milan Dataset	44
3.4.3	Gray-level Descriptors	47
3.4.4	Color Descriptors	48
4	Convolutional Neural Networks for Object Recognition	57
4.1	Multilayer Perceptron	58
4.2	Convolutional Neural Networks	59
4.3	Training a Neural Network	60
4.3.1	Gradient Descent	60
4.3.2	Stochastic Gradient Descent	61
4.3.3	Momentum and other variants	61
4.4	Data preprocessing	63
4.5	Common CNN Layers	64
4.6	State-of-the-art CNN architectures	67
4.6.1	AlexNet (2012)	68
4.6.2	VGG (2014)	69
4.6.3	Inception (2014)	69
4.6.4	Resnet (2015)	70
4.7	Network Parameters	72
4.8	Data Augmentation	73
4.9	Regularization	74
5	Logo Recognition	79
5.1	Datasets	80
5.1.1	FlickrLogos-32 Dataset	80
5.1.2	Logos-32plus Dataset	80
5.1.3	Duplicates Removal	82
5.2	Processing Pipeline	83
5.3	Selective Search	84
5.3.1	Evaluation on Flickr-32 dataset	85

5.4	Query expansion	86
5.5	CNN features	86
5.6	Transformation Pursuit	87
5.7	Experimental Setup and Results	87
5.7.1	Robustness to Image Distortions	88
5.7.2	Selective Search Evaluation	90
5.8	Improved pipeline	91
5.8.1	Network Architecture	94
5.9	Experimental Setup and Results	95
5.10	Timings	98
6	Painting Categorization	101
6.1	Related works	102
6.2	Proposed Approach	102
6.2.1	Input preprocessing	103
6.2.2	Spatial Transformer Network	103
6.2.3	Deep Network Structure	104
6.2.4	Hand-crafted feature injection	106
6.3	Datasets	107
6.3.1	Painting91	107
6.3.2	Wikipaintings-IVL Dataset	107
6.3.3	Synthetic Data Augmentation	108
6.4	Experiments	109
6.4.1	Training	109
6.4.2	Results on Painting-91 dataset	110
6.4.3	Multitask and Spatial Transformer Network	111
6.4.4	Synthetic Data Augmentation	114
6.4.5	Handcrafted descriptors	114
6.5	Similarity search	116
7	Conclusions	121
	References	125
	Appendix A Texture Synthesis and Style Transfer	137
A.1	Texture synthesis	137
A.2	Image Style Transfer	140
A.3	Feed-forward network and Perceptual loss	142

List of figures

2.1	Light intensity change	8
2.2	Light intensity shift	8
2.3	Light color change	9
2.4	Change of camera viewpoint	12
2.5	Perspective handling	13
2.6	Blur Noise and JPEG compression effects	14
3.1	Harris good interest points	17
3.2	Moravec Window Calculation	18
3.3	Harris Eigenvalues Distribution	19
3.4	Difference Of Gaussians	23
3.5	Fast-Hessian derivatives approximation	25
3.6	FAST Corner detection	26
3.7	Kaze scalespace	28
3.8	SIFT descriptor sampling structure	29
3.9	SURF descriptor	30
3.10	FREAK sampling pattern	31
3.11	Cans of tea lemon and peach-flavoured	32
3.12	Late vs early color fusion approaches.	35
3.13	Pairwise matching experiment evaluation Pipeline	38
3.14	Typical object from the CDVS dataset.	40
3.15	Images from Turin dataset.	40
3.16	Images from Common objects and scenes dataset.	41
3.17	Images from SuperMarket Milan dataset.	42
3.18	Examples of color significant products.	43
3.19	TPR, FPR, MAP and Top Match measures on all datasets.	46
3.20	TPR for gray-scale algorithms. No limit on keypoints number.	49
3.21	TPR for gray-scale algorithms. Keypoints number limit.	50

List of figures

3.22	TPR for color algorithms. No limit on keypoints number.	53
3.23	TPR for color algorithms. Limited keypoints number.	54
3.24	Examples of images where color information can be misleading.	55
4.1	Perceptron	58
4.2	Effect of changing learning rate.	60
4.3	Decorrelation	63
4.4	CNN layer	64
4.5	Non linear activation functions	66
4.6	AlexNet CNN architecture.	68
4.7	Inception module.	70
4.8	Resnet block	71
4.9	Deep CNN Params vs Accuracy	72
5.1	FlickrLogos-32 dataset example images	80
5.2	Queries composition used to download Logos-32plus dataset.	81
5.3	Example of near duplicates.	83
5.4	Distribution of FlickrLogos-32 and Logos-32plus dataset.	84
5.5	Recognition pipeline outline.	84
5.6	Color spaces comparison for object proposal on Flickr-Logos dataset.	85
5.7	Geometric transformations applied to a region proposal.	87
5.8	Precision-Recall curve obtained with Query Expansion.	89
5.9	Types of distortions applied to the images of the FlickrLogos-32 dataset.	90
5.10	Recal vs IoU threshold on Flickr-Logos dataset.	90
5.11	Precision-Recall curves for different image distortions.	91
5.12	Simplified logo classification pipeline	92
5.13	Logo recognition training framework.	94
5.14	Logo recognition testing framework.	94
5.15	Wrongly labeled logos.	98
6.1	Scheme of the Deep Multibranch Multitask Neural Network.	104
6.2	Deep CNN Residual Blocks	106
6.3	Paintings from Wikipaintings-IVL dataset	108
6.4	Examples of real and synthetic paintings	109
6.5	Confusion matrix for the task of style recognition	112
6.6	Confusion matrix for the task of artist recognition	113
6.7	Highest scored errors for the task of painters classification	113
6.8	Classification accuracy using handcrafted features	117

6.9	Percentage of correctly classified examples	118
6.10	Screen-shot of the proposed similarity search system.	120
A.1	Examples of synthetic textures	137
A.2	Texture Synthesis Network	138
A.3	Example of texture synthesis output.	139
A.4	Style Transfer Network	140
A.5	Visual results of tuning Style Transfer Network hyperparameters.	144
A.6	Style Transfer Network	145
A.7	Examples of stylized images produced by the feedforward CNN	145

List of tables

3.1	Datasets adopted in the MPEG CDVS test model	39
3.2	Gray-levels algorithms investigated	44
3.3	Evaluated color descriptors	45
3.4	Detectors and descriptors performances comparison	51
5.1	Comparison between FlickrLogos-32 and Logos-32plus datasets	83
5.2	Color space comparison for object proposal.	86
5.3	FlickrLogos-32 dataset partitions	88
5.4	Performance comparison with other approaches	89
5.5	FlickrLogos-32 dataset image distortions.	89
5.6	Convolutional Neural Network Architecture	95
5.7	Results on different training choices.	96
5.8	Comparison with state-of-the-art.	99
5.9	Recognition pipeline timings.	99
6.1	Structure of the Multibranch Multitask Deep Neural Network.	105
6.2	Paintings-91 vs Wikipaintings-IVL.	108
6.3	Accuracy vs Number of Passes trough the Network	110
6.4	Comparison with the state of the art	111
6.5	Classification accuracy on Wikipaintings-IVL dataset	114
6.6	Accuracy measures obtained using synthetic data	114
6.7	Classification accuracy on Wikipaintings-IVL dataset	116

Chapter 1

Introduction

“I can see the cup on the table,” interrupted Diogenes, “but I can’t see the cupness.” “That’s because you have the eyes to see the cup,” said Plato, “but”, tapping his head with his forefinger, “you don’t have the intellect with which to comprehend cupness.”

Teachings of Diogenes

It has been estimated that humans can distinguish on the order of 30,000 visual categories [117]. They also have a remarkable aptitude to remember pictures [63]. E.g. Grady et al. [52], in their experiments, discovered that people can remember more than 2000 pictures with at least 90% of accuracy in recognition tests over several days. Excellent memory for pictures consistently exceeds our capacity to remember words. In the brain itself, neurons dedicated to visual processing are hundreds of millions and engage about 30% of the cortex, compared with 8% for touch and 3% for hearing. This underlines the importance of the vision system in human brain and the nature of complexity of visual signals coming from a real-world experience.

Human brain has to deal with a huge amount of environment and light conditions such as changes in light color and intensity, drastic differences in viewpoints, partial occlusions among others. Nonetheless it has the ability to codify equivariant representations of objects and scenes. The role of a feature extractor is somehow similar to the role of the inferior temporal cortex in primates, i.e. to codify the appearance of a visual concept such that the description is equivariant/invariant to different conditions and at the same time is highly discriminative with respect to precise tasks.

In computer vision, one form of feature extraction is concerned with the detection and description of important image regions. Approaches adopting this paradigm are generally referred to as *Keypoint-based*. They fall into a broader category of approaches named *Handcrafted* as opposite to those based on *Learned features*. Recently, the

Introduction

reintroduction of Neural Networks into many computer vision tasks broadly integrated handcrafted approaches. Neural networks learn feature extraction as part of an end-to-end pipeline. Hence, learned features are tailored on data and the Network is able to produce complex features hierarchies that are highly discriminative. While these approaches have shown great success in tasks such as scene recognition, object detection and classification, other tasks such as structure-from-motion still depend on purely engineered features to detect and describe keypoints. Moreover the success of Neural Networks in the last years has been made possible by the exponential increase of computational power and the extensive use of GPUs to compute parallel matricial operations. However GPUs are still very expensive. Furthermore the power consumption and miniaturization haven't yet reached the right maturity to allow the plain use of Neural Networks on embedded devices. Mobile phones and widespread mobile devices do not have the capability to handle heavy burden computations needed by Neural Networks. Thus, for a large variety of applications, Keypoint-based approaches represent one of the most affordable and reliable tools for object and scene recognition.

Newest mobile phones make use of object recognition techniques to perform visual search for an enhanced user experience. As an example the new Pixel 2 XL from Google and the new Huawei models employ visual search as a standard component of the operative system, providing new ways to search for multimedia contents. As a matter of fact, such applications rely on strong feature extractors and good and general purpose object detectors. The employment of these technologies opens a variety of applications on different domains. In the retail market a user can take a picture of an object on a store shelf and get information about that product. A set of feature detectors and extractors is evaluated on this type of task in Section 3.3. Performances are accessed in real-world acquisition conditions with different categories of objects, from buildings to common objects to CDs, DVDs and supermarket customer goods. A similar application but in a different domain is introduced in Section 6 where a painting categorization system is presented. The use of such system can be used as a useful tool in museums or on consumer mobile applications.

Tracking a particular brand trough social networks can be a unique source of information to estimate a general trend, to track changes in market segments or to discover user sentiment and preferences. Such tasks need strong object detectors to find brands. Section 5 presents a logo recognition model that can be used for this type of applications. Real-world pictures from Flickr are used for training and testing.

Augmented Reality applications rely on good feature extractors to recognize scenes and potentially to track objects. Traditionally Keypoint-based approaches are employed, being reliable and computationally efficient for the use on embedded and mobile devices. In such applications usually actions are triggered by some real-world events or after the detection of a particular object. Thus Augmented Reality, in most cases, involves Visual Search. The idea is to allow the software to interact with real objects in a marker less environment. Algorithms developed in Section 6 for Painting Recognition can be used for cultural and didactic programs where informations about a specific painting is shown and the user explores interactive environments. Similar applications can be built by exploiting algorithms introduced in Section 3.3 for object recognition or those in Section 5 for logo recognition.

1.1 Thesis overview

In this thesis an in-depth study of detectors and descriptors for object recognition is presented. Chapter 2 introduces local image transformations. A reflectance model is presented and then the most common transformations are introduced within this framework. Chapter 3 is a review of the state-of-the-art Interest Point Detectors and Descriptors. It includes an experimental Section that shows a comparison of these algorithms. In the same chapter the standard CDVS Test Model Framework is presented. The results of different Pairwise Matching Experiments are shown on different datasets. Keypoint-based methods reported high performances in term of True Positive Rate vs False Positive Rate and are proven to be robust for real-world applications. Then, in order to test descriptors on a new use case, a dataset named Supermarket Milan has been introduced. State-of-the-art descriptors are blind to color and does not have the capability to discriminate *Color Significant Products* introduced in Section 3.3.4. To extend their domain of applicability they have been enriched to handle color information and tested on the new dataset. Chapter 4 introduces Convolutional Neural Networks applied to object recognition. A list of well-known architectures is presented together with some related arguments: data preprocessing, data augmentation and regularization. In Chapter 5 and 6 Convolutional Neural Networks are applied to object recognition. In particular, two applications are investigated: Logo Recognition and Painting Categorization. Chapter 5 concerns Logo Recognition introduces a two stage processing pipeline to allow the use of Convolutional Neural Networks on high resolution images. Two types of networks and their application to the Flickr32 challenging dataset are investigated. A quantitative comparison between keypoint-

Introduction

based approaches and Neural Networks is shown where Neural Networks outperform keypoint-based approaches especially when dealing with low-quality, blurred or noisy images. Chapter 6 regards the use of Convolutional Networks for the task of Painting Categorization to predict author, style and genre of paintings. A new Deep CNN structure is proposed. The joint use of handcrafted and neural features is investigated in order to improve classification accuracy. Different experiments are made on a challenging dataset with more than 1500 painters. Finally, Chapter 7 ends the thesis summarizing the results obtained, reporting the conclusions and giving the directions for future works.

Chapter 2

Common Local Image Transformations

Features extractors have been engineered to produce descriptions with invariance properties to certain types of image transformations. The type of invariance depends on the application, and consequently, strongly on the image acquisition conditions.

As an example, for a quality control application in a factory, it is usually possible to control environment variables during the time of the image acquisition e.g. to set a fixed light intensity in the scene. It may be also possible to fasten the camera somewhere at a precise distance in front of the object to be acquired. In this case it is easier to calculate some physical properties of the object, like size. The detector/descriptor hasn't to be invariant to illumination conditions and has not to be invariant to scale transformations.

There are other real cases in which these properties are expected to hold. For example a mosaicing application. Outdoor pictures may have different illumination conditions and the same objects depicted could likely be at slightly different scales.

This section presents an overview of the typical local image transformations that can occur in real cases. Most of the detectors and descriptors presented in the next chapters can handle these types of transformations.

2.1 Light change

In the following sections the Lambertian reflectance model and the Von Kries illumination model are introduced. Then five lighting transformations are presented. The notation used follows closely the one adopted by van de Sande et al. in [135].

2.1.1 Reflectance model

In this section the *Lambertian reflectance model* is introduced. It represents an ideal reflecting surface where the luminance is uniform in all orientations, i.e. it is isotropic, and obeys to the Lambert's cosine law.

Under this assumptions, image pixels can be modeled as follows:

$$\mathbf{f}(\mathbf{x}) = \int_{\omega} e(\lambda)\rho_k(\lambda)s(\mathbf{x}, \lambda)d\lambda + \int_{\omega} A(\lambda)\rho_k(\lambda)d\lambda \quad (2.1)$$

where $e(\lambda)$ represent the color of the light source, $\rho_k(\lambda)$ is the camera sensitivity function with $k \in \{R, G, B\}$, $s(\mathbf{x}, \lambda)$ is the surface reflectance and $A(\lambda)$ represents the diffuse light. Furthermore ω is the visible spectrum and \mathbf{x} represents the vector of spatial coordinates.

By deriving the reflection model, we can see that the equation term $A(\lambda)$, which represents the diffuse light, is cancelled out, because it is independent of the surface reflectance term. The final derivative equation is:

$$\mathbf{f}_{x,\sigma}(\mathbf{x}) = \int_{\omega} e(\lambda)\rho_k(\lambda)s_{x,\sigma}(\mathbf{x}, \lambda)d\lambda \quad (2.2)$$

Removal of the $A(\lambda)$ term is an important result because we have achieved a first type of invariance: descriptors based on derivatives will yield invariance to diffuse light.

2.1.2 Von Kries Model

In this section the *Von Kries Model*, also called *Diagonal Model* is introduced. Conceived by Von Kries in the 1970 [142], this is a simple but significant model used to describe illumination changes under the assumption of narrow band filters.

Image acquired under an unknown light source \mathbf{f}^u is transformed with a diagonal matrix $\mathcal{D}^{u,c}$ to \mathbf{f}^c , that is the same image transformed, as if it was taken under the reference light, also named the *canonical illuminant*. This is the mapping function:

$$\mathbf{f}^c = \mathcal{D}^{u,c}\mathbf{f}^u \quad (2.3)$$

The equivalent matricial form of the equation is the following:

$$\begin{pmatrix} R^c \\ G^c \\ B^c \end{pmatrix} = \begin{pmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & c \end{pmatrix} \begin{pmatrix} R^u \\ G^u \\ B^u \end{pmatrix} \quad (2.4)$$

adding the diffuse light term, the resulting diagonal model became:

$$\begin{pmatrix} R^c \\ G^c \\ B^c \end{pmatrix} = \begin{pmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & c \end{pmatrix} \begin{pmatrix} R^u \\ G^u \\ B^u \end{pmatrix} + \begin{pmatrix} O_1 \\ O_2 \\ O_3 \end{pmatrix} \quad (2.5)$$

The surface reflectance $s(\mathbf{x}, \lambda_c)$ of the equation (2.1) is equal for both the canonical and the unknown illuminant, so this term can be cancelled out. After this, the equation (2.5) becomes:

$$\begin{pmatrix} e^c(\lambda_R) \\ e^c(\lambda_G) \\ e^c(\lambda_B) \end{pmatrix} = \begin{pmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & c \end{pmatrix} \begin{pmatrix} e^u(\lambda_R) \\ e^u(\lambda_G) \\ e^u(\lambda_B) \end{pmatrix} + \begin{pmatrix} A(\lambda_R) \\ A(\lambda_G) \\ A(\lambda_B) \end{pmatrix} \quad (2.6)$$

The next section shows five lighting transformations based on these two models presented above.

2.1.3 Five lighting transformations

Light intensity change image values are changed by a constant factor in all the three channels ($a = b = c$):

$$\begin{pmatrix} R^c \\ G^c \\ B^c \end{pmatrix} = \begin{pmatrix} a & 0 & 0 \\ 0 & a & 0 \\ 0 & 0 & a \end{pmatrix} \begin{pmatrix} R^u \\ G^u \\ B^u \end{pmatrix} \quad (2.7)$$

Note that light intensity changes also include shadows and shading. When a detector or descriptor is invariant to light intensity changes, it is *scale-invariant* with respect to light intensity.

Light intensity shift image values change by an equal offset in all channels ($a = b = c = 1, O_1 = O_2 = O_3$):

$$\begin{pmatrix} R^c \\ G^c \\ B^c \end{pmatrix} = \begin{pmatrix} R^u \\ G^u \\ B^u \end{pmatrix} + \begin{pmatrix} O_1 \\ O_1 \\ O_1 \end{pmatrix} \quad (2.8)$$

Light intensity shifts are models of real phenomenon like object highlights under a white light source, interreflections, scattering of a white light source, and infrared

Common Local Image Transformations

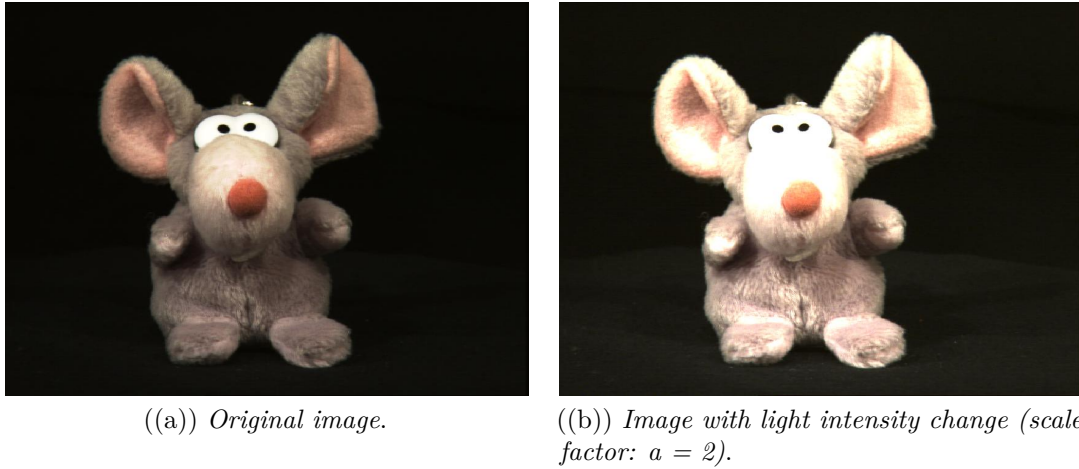


Fig. 2.1 Example of light intensity change transformation. Images from ALOI [48] dataset.

sensitivity of the camera sensor. When a detector or a descriptor is invariant to a light intensity shift, it is *shift-invariant* with respect to light intensity.

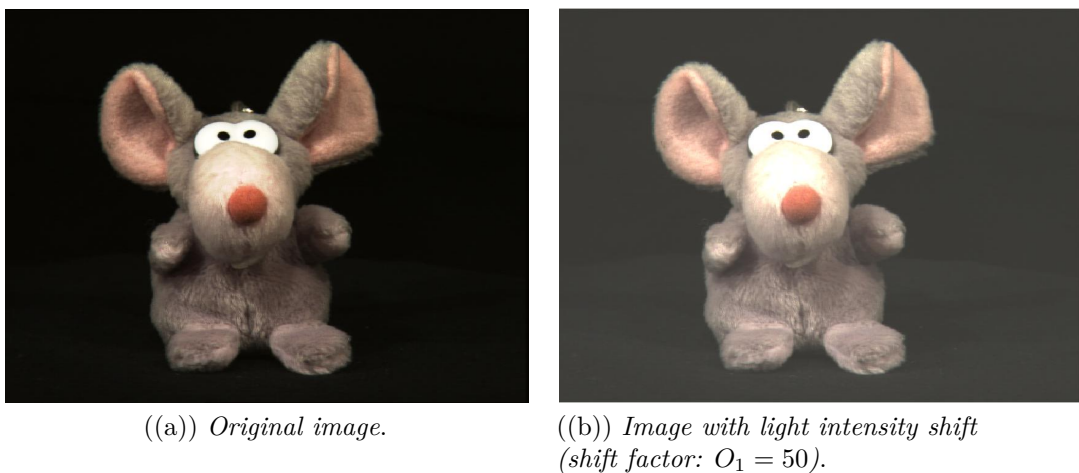


Fig. 2.2 Example of light intensity shift. Original image from ALOI [48] dataset.

Light intensity change and shift Image values change by combining the two kinds of change above:

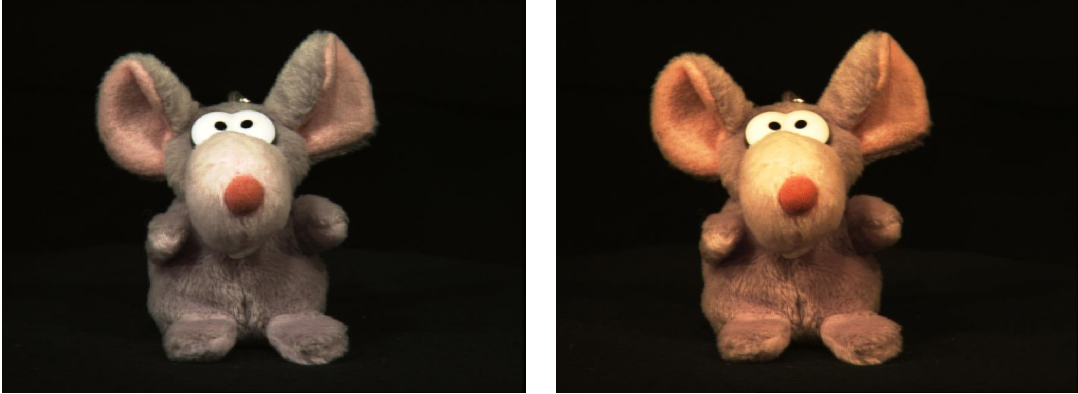
$$\begin{pmatrix} R^c \\ G^c \\ B^c \end{pmatrix} = \begin{pmatrix} a & 0 & 0 \\ 0 & a & 0 \\ 0 & 0 & a \end{pmatrix} \begin{pmatrix} R^u \\ G^u \\ B^u \end{pmatrix} + \begin{pmatrix} O_1 \\ O_1 \\ O_1 \end{pmatrix} \quad (2.9)$$

A detector/descriptor invariant to these changes is called scale-invariant and shift-invariant with respect to light intensity.

Light color change Image values change in all channels independently ($a \neq b \neq c$):

$$\begin{pmatrix} R^c \\ G^c \\ B^c \end{pmatrix} = \begin{pmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & c \end{pmatrix} \begin{pmatrix} R^u \\ G^u \\ B^u \end{pmatrix} \quad (2.10)$$

This kind of changes can model a change in the illuminant color.



((a)) *Original image.*

((b)) *Image with color change.*

Fig. 2.3 Example of light color change. Right image appears warmer. That is because of a yellow illuminant. Images from ALOI [48] dataset.

Light color change and shift Image values change in all channels independently ($a \neq b \neq c$) with arbitrary offsets ($O_1 \neq O_2 \neq O_3$). This is the expression of the full diagonal model of the equation (2.5) :

$$\begin{pmatrix} R^c \\ G^c \\ B^c \end{pmatrix} = \begin{pmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & c \end{pmatrix} \begin{pmatrix} R^u \\ G^u \\ B^u \end{pmatrix} + \begin{pmatrix} O_1 \\ O_2 \\ O_3 \end{pmatrix} \quad (2.11)$$

Five types of common changes have been identified based on the diagonal-offset model of illumination change.

The type of light transformations discussed in this chapter are mentioned as light scale and shift invariance properties for detectors/descriptors.

2.2 Affine transformations

Affine transformations are a widely-used geometric transformation model because of their simple form, involving few parameters, being a good approximation of more complex transformations when dealing with local image patches.

Affine transformations preserve collinearity, i.e., all points lying on a line initially still lie on a line after transformation, and ratios of distances, e.g., the midpoint of a line segment remains the midpoint after transformation.

In general, affine transformations are compositions of rotations, translations, dilations, and shears. Furthermore, affine transformation can be expressed in the form of a matrix multiplication or linear transformation followed by a vector addition i.e. translation. Equation 2.12 represents an affine transformation in matrix form.

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{bmatrix} \begin{bmatrix} x' \\ y' \end{bmatrix} + \begin{bmatrix} b_0 \\ b_1 \end{bmatrix} \quad (2.12)$$

Further, it's possible to join the multiplication matrix and the translation vector into a single matrix by converting them into the homogeneous coordinate system, the resulting matrix is squared and invertible:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} a_{00} & a_{01} & b_0 \\ a_{10} & a_{11} & b_1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \quad (2.13)$$

Here is shown the composition of the affine transformation matrix:

$$\begin{bmatrix} a_{00} & a_{01} & b_0 \\ a_{10} & a_{11} & b_1 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \alpha & \beta & (1 - \alpha) \cdot c(x) - \beta \cdot c(y) \\ -\beta & \alpha & \beta \cdot c(x) + (1 - \alpha) \cdot c(y) \\ 0 & 0 & 1 \end{bmatrix} \quad (2.14)$$

where $\alpha = scale \cdot \cos(\theta)$ and $\beta = scale \cdot \sin(\theta)$. θ is the rotation angle, $scale$ is the isotropic scale factor, and $c(x), c(y)$ represent the x-coordinate and y-coordinate of the rotation center respectively.

Given these parameters it is possible to obtain any type of simple or composed affine transformation.

Camera point of view change: In a common real-world application it is likely to deal with 3D geometric transformations caused by the change of camera point of

view. In Fig. 2.4 two matching areas of the same graffiti lies on two different images taken at different point of view. This is not an in-plane transformation but it is a 3D out-of-plane transformation. However it can be approximately handled by an affine transformation. Photos are from Mikolajczyk dataset [96] thus the transformation between the two images is well known. Here is the *homography* matrix of the two images, where homography means an affine transformation referred to a projective plane.

$$\begin{bmatrix} 0.625 & 0.057 & 222 \\ 0.222 & 1.165 & -25.606 \\ 0.000 & 0.000 & 1.000 \end{bmatrix} \quad (2.15)$$

Via the QR-decomposition process it is possible to decompose the homography matrix H into two matrices Q and R where Q is an orthogonal matrix $Q^T Q = I$ and R is an upper triangular matrix:

$$Q = \begin{bmatrix} -0.9422 & 0.3350 & -0.0008 \\ -0.3350 & -0.9422 & 0.0001 \\ -0.0007 & 0.0003 & 1.0000 \end{bmatrix} \quad (2.16)$$

$$R = \begin{bmatrix} -0.6638 & -0.4448 & -200.6023 \\ 0 & -1.0785 & 98.5094 \\ 0 & 0 & 0.8178 \end{bmatrix}$$

Now geometric properties of the transformation are decomposed. Here is shown where they are located:

$$Q = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad R = \begin{bmatrix} s_x & s_h & t_x \\ 0 & s_y & t_y \\ 0 & 0 & 1 \end{bmatrix} \quad (2.17)$$

Q matrix contains all the rotation informations and the R matrix contains scale, translation and shear. t_x and t_y represent the shift factors, s_x and s_y the scale and s_h the shear (i.e. ratio between unit of measurement of the scales of the two axes).

Perspective lines: Black lines in Fig. 2.5 highlight the typical *accidental perspective* effect. All the horizontal lines are no more parallel like in the frontal image but they converge in a perspective vanishing point.

Since affine transformations include only geometric changes in which parallels lines

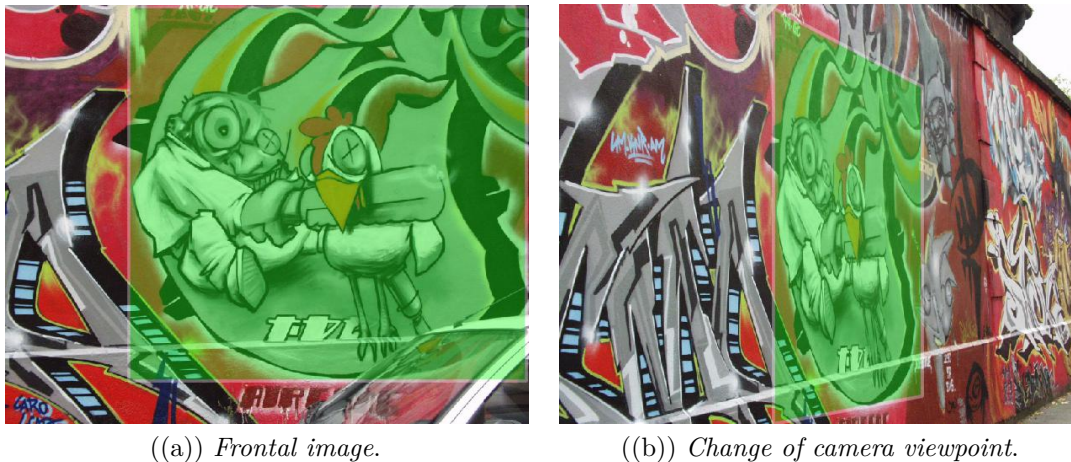


Fig. 2.4 Example of the same scene from different viewpoints. These 3D geometric transformations can be approximated by an affine transformation (2D). Images from [95].

remain parallels, this kind of transformations can't be handled. For this reason, algorithms considering affine transformations can only provide approximated solutions.

2.3 Blur, Noise and JPEG compression

Blur, noise and JPEG compression have a similar effect on images from a signal-processing point-of-view. They cause information loss and partially modify fine image details.

Therefore a good descriptor should build a good representation of the image coarse structure with less relevance on the finest details. In Figure 2.6 the upper-right image is compressed with the JPEG algorithm. The most clear artifact can be found in particular on patches with high frequencies and it is related to the appearance of pixels blocks. The lower-left image presents a blurring transformation. The only noticeable effect is the loss of finest details, i.e. high frequencies, this is accentuated particularly on the trees.

The lower-right image is the one with Gaussian noise added. The visual effect is the opposite of the previous transformations. Random noise is added to the image substituting useful information, this transformation induces an information loss.



Fig. 2.5 Black lines showing perspective vanishing point. Original image from [95].

Common Local Image Transformations



((a)) *Original image.*



((b)) *High JPEG compression.*



((c)) *Image blurred with a Gaussian filter (window size=7x7 sigma=5).*



((d)) *Image corrupted with Gaussian noise (level=0.01) on the three RGB channels independently.*

Fig. 2.6 Examples showing effects of Blur, Noise and JPEG compression. Original and JPEG images from Mikolajczyk [95].

Chapter 3

Keypoint-based Approaches

In this chapter a qualitative and quantitative analysis of methods to perform Instance Recognition is presented. The analysis focuses on keypoint-based detectors and descriptors for local interest regions. First, an overview of the common local image transformations is made. Local detectors and descriptors should be robust with respect to this types of transformations. Then different algorithms in the state-of-the-art are described and finally extensive quantitative test are made on a set of standard datasets.

3.1 Interest Point Detectors

The term *Point of Interest* has been introduced by Moravec [97] for a subclass of detectors using autocorrelation methods but, over time, this term became the term to describe entire class of local detectors algorithms.

There are mainly two class of detectors. Blob detectors and Corner detectors. The difference between them is the kind of image patch they detect:

- **Blob detectors:** algorithms that search and extract blob regions. Keypoints (blobs) are the centers of elliptic areas, with a particular variation of graylevel, compared to the region around them.
- **Corner detectors:** these kind of algorithms search and extract image patches of objects' corners.

3.1.1 Desirable properties of corner detectors

D. Parks and J.P. Gravel in [103] describe a list of properties that a corner detector must have. This list is extensible to a generic Interest Point Detector:

Keypoint-based Approaches

1. All *true corners* should be detected.
2. No *false corners* should be detected.
3. Corner points should be well localized.
4. Detector should have a high repeatability rate, i.e. good stability.
5. Detector should be robust with respect to noise.
6. Detector should be computationally efficient.

There is no objective definition for what a *true corner* is and what a false corner is. It depends most on the application the detector is used for. However, a good corner is somehow different from an edge. This has to do with the fourth property: a corner is generically an image patch where the intensity variation is at least bi-directional while the variation of an edge could be one-directional. For that reason a corner usually have better stability than any edge.

Localization refers to how accurately the position of a corner is found, good localization it is not critical for all applications but obviously a descriptor produces better descriptions with better localization of the detector.

A corner detector must be robust even against noise because it is unavoidable in many applications. Computational efficiency is a surplus for all applications but it is a must for real-time applications. That is why for this kind of applications is often preferred the use of a less accurate but faster detector.

3.1.2 Harris detector

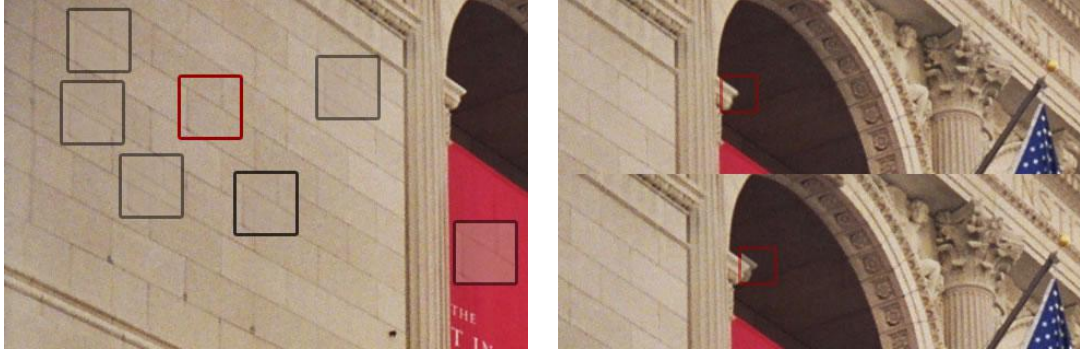
This corner detector was developed by Chris Harris and Mike Stephens in 1988 [56]. The 1988 version of the algorithm over time it is been improved and nowadays many detectors are based on the underlying ideas of the original Harris detector. Harris and Stephens made this detector by addressing the limitations of the older Moravec detector [97]. Despite an increase of computational time compared to Moravec, it has a significantly higher repeatability rate.

Harris is part of a subclass of corner detectors called *Autocorrelation methods*. It has been conceived to find interest points that generate large variations when moved around the image.

Figure (3.1) shows the underlying idea. The right image shows a bad interest point inside the red bounding-box. A pixel comparison with the others patches inside the

gray windows shows a very low difference.

The up-right image shows a good corner point. The down-right image shows that a little change of position originate large variations. The most intuitive way to find



((a)) *Bad interest point.*

((b)) *Good interest point.*

Fig. 3.1 Harris autocorrelation method. Bad interest point vs. good interest point. Images source www.aishack.in [Sinha].

image patches that produce the highest variations is move a window in every direction and compare the obtained window with the original one doing the sum of the squared differences between each pixel. This is the method used by the Moravec operator. The idea underlying the Harris detector is an approximation of the Moravec operator which is made by using the image gradient.

Equation 3.1 is a mathematical formulation of the difference between the original and the moved window. Localizing corners can be formulated as a search for maxima of this function.

$$V(u, v) = \sum_{x,y} w(x, y) [I(x + u, y + v) - I(x, y)]^2 \quad (3.1)$$

(u, v) are the window's displacements in the x and y directions respectively. $w(x, y)$ is the window at position (x, y) . It acts like a mask, ensuring that only the desired window is used. I is the intensity of the image at position (x, y) .

Equation 3.1 is then approximated using the first term of the Taylor expansion 3.2:

$$V(u, v) \approx \sum_{x,y} [I(x, y) + uI_x + vI_y - I(x, y)]^2 \quad (3.2)$$

The next step is the square expansion in Equation 3.3:

$$V(u, v) \approx \sum_{x,y} u^2 I_x^2 + 2uv I_x I_y + v^2 I_y^2 \quad (3.3)$$

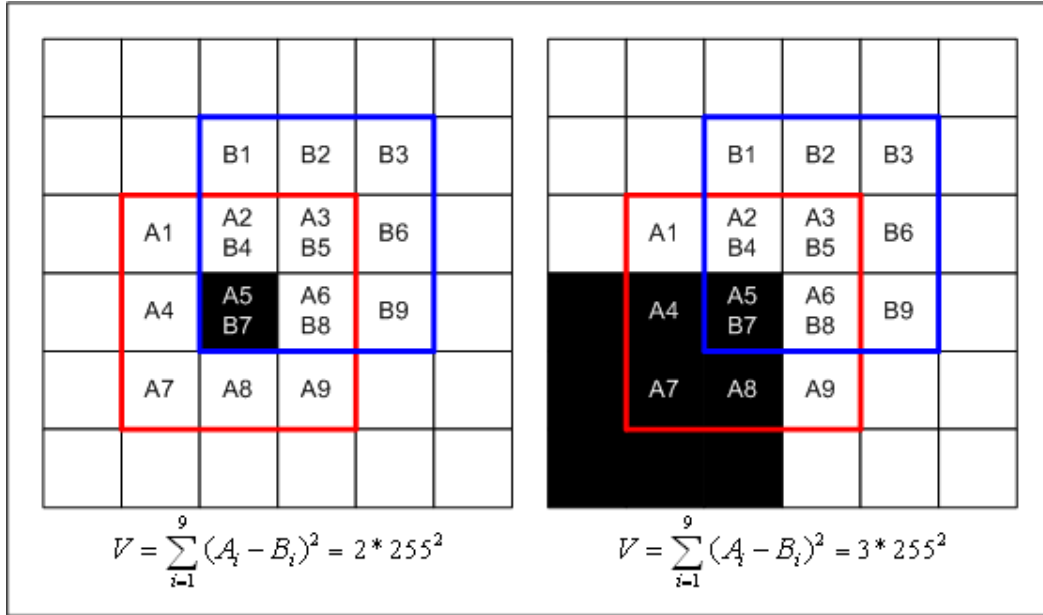


Fig. 3.2 Example of Moravec Window Calculation. Image from [103].

The final intuition is that this equation can be written as a matrix equation (Equation 3.4):

$$V(u, v) \approx \begin{bmatrix} u & v \end{bmatrix} \left(\sum \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \right) \begin{bmatrix} u \\ v \end{bmatrix} \quad (3.4)$$

the obtained matrix is called M (with the $w(x, y)$ term added).

$$M = \sum w(x, y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} = \begin{bmatrix} A & C \\ C & B \end{bmatrix} \quad (3.5)$$

Equation 3.6 represents a compact view of Equation 3.8:

$$V(u, v) \approx \begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix} \quad (3.6)$$

Note that matrix M contains all the differential operators describing the geometry of the image surface at a given point (x, y) . Harris and Stephens [56] proposed a

Cornerness Measure based on the eigenvalues of this matrix:

$$\begin{aligned}
 C(x, y) &= \det(M) - k(\text{trace}(M))^2 \\
 \det(M) &= \lambda_1 \lambda_2 = AB - C^2 \\
 \text{trace}(M) &= \lambda_1 + \lambda_2 = A + B \\
 k &= \text{constant}
 \end{aligned}
 \tag{3.7}$$

Possible values of k are $k = 0.2$, $k = 0.1$ or $k = 0.05$.

Image (3.3) shows a typical distribution of the two eigenvalues of the M matrix. α and β correspond to λ_1 and λ_2 with notation used in this thesis.

The edge regions are noticeable near the two axes. The flat region near the origin,

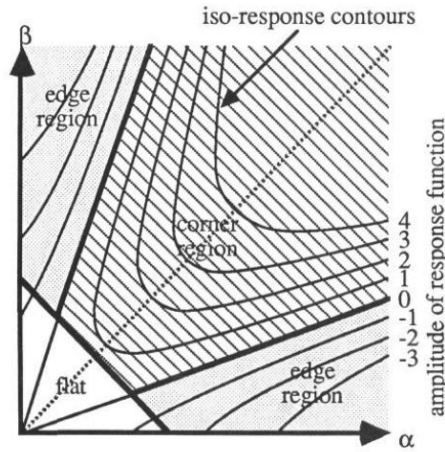


Fig. 3.3 Auto-correlation principal curvature space. Heavy lines give corner/edge/flat classification, fine lines are equi-response contours. Figure from [56].

and the corner region far from the axes.

If both eigenvalues are small, the local autocorrelation function is flat and the windowed image region is of approximately constant intensity. If one eigenvalue is high and the other low we are talking about an edge. If both eigenvalues are high, this indicate the presence of a corner.

Changing the values of k brings to a transformation of the function. For smaller values of k the corner region increases while for bigger values it decreases. $C(x, y)$ is a function which value increases if the curvature of the region are near the center of the corner region.

3.1.3 Scale and affine invariant harris detector

In 2004 Krystian Mikolajczyk and Cordelia Schmid proposed a novel method to perform keypoint detection [92] [94] that can deal with significant affine transformations including large scale changes.

Their first work was based on a modified version of the Harris detector, a second version of the algorithm was instead based on the Hessian cornerness measure. In Section 3.4 it will be presented an evaluation of performances of the Mikolajczyk Affine Detector with these three cornerness measures: Harris, Hessian and DoG.

Scale Invariance To achieve scale invariance property they made modifications to the second moment matrix used for detection:

$$\mu(\mathbf{x}, \sigma_I, \sigma_D) = \begin{bmatrix} \mu_{11} & \mu_{12} \\ \mu_{21} & \mu_{22} \end{bmatrix} = \sigma_D^2 g(\sigma_I) \begin{bmatrix} L_x^2(\mathbf{x}, \sigma_D) & L_x L_y(\mathbf{x}, \sigma_D) \\ L_x L_y(\mathbf{x}, \sigma_D) & L_y^2(\mathbf{x}, \sigma_D) \end{bmatrix} \quad (3.8)$$

The second moment matrix now depends on σ_I and σ_D which represent the integration and differentiation scales respectively. L_x and L_y are the derivatives in the x and y directions respectively. The local derivatives are computed with Gaussian kernels of the size determined by the local scale σ_D (differentiation scale). The derivatives are then averaged in the neighborhood of the point by smoothing with a Gaussian windows of size σ_I (integration scale).

The Harris cornerness measure is also modified to fulfill the changes on the second moment matrix:

$$C = \det(\mu(x, \sigma_I, \sigma_D) - k \text{trace}^2(\mu(x, \sigma_I, \sigma_D))) \quad (3.9)$$

Local maxima of the C function determine the location of interest points.

Given an interest point in scale-space it is possible to perform automatic scale selection. The underlying idea is to select for each keypoint the characteristic scale of a local structure, i.e. the scale where the filter attains a maximum.

After various experiments, Mikolajczyk and Schmid noticed that Harris measure doesn't fit well this task, so they adopted Laplacian-of-Gaussians for the scale selection because of its ability to individuate blob-like structure.

The final algorithm consists of three steps: a point detection over all the scales and an iterative selection of the scale and the location of each point. The first step is performed with the Harris filter for a set of pre-selected scale measures. The second

step is made by an iterative algorithm that simultaneously detect the maximum over scales and refine the keypoint location as follows:

1. Find the local extremum over scale of the LoG for the given point.
2. Detect the spatial location of a maximum of the Harris measure nearest to the point at the found scale.
3. Go to step 1 if the scale or the point location changes are over a given threshold.

Affine Invariance To achieve affine invariance the main idea, first explored by Lindeberg [81], is to use the second moment matrix as an estimator for the anisotropic shape of a local image structure.

Without loss of generality it is right to suppose a local anisotropic structure as an affine transformed isotropic structure. Therefore the output of the Harris detector fits well with this approach. A new measure is introduced:

$$Q = \frac{\lambda_{min}(\mu)}{\lambda_{max}(\mu)} \quad (3.10)$$

where λ_{min} and λ_{max} are the second moment matrix eigenvalues. Q represents the ratio between the eigenvalues which is high for anisotropic image patches and low for isotropic patches.

After the first step in which keypoints are detected in the multiscale space with Harris filter, every point location and shape is refined by an iterative algorithm. The procedure used is similar as for the Automatic Scale Detection, but with a difference. At every iteration step, it is computed an affine transformation on the neighborhood of the keypoint. Therefore the convergence criterion is based on the eigenvalues ratio.

3.1.4 SIFT Detector

Published by David Lowe in 1999 [82], SIFT is the acronym for Scale Invariant Feature Transform. Unlike the others detectors presented in this chapter, SIFT algorithm, is patented in the US by the University of British Columbia, in particular the step of the pipeline that involves Difference of Gaussian for keypoints detection.

It is composed by three stages:

1. *Scale-space extrema detection* The first stage of computation searches over all scales and image locations. It is implemented efficiently by using a difference-of-

Keypoint-based Approaches

Gaussian function to identify potential interest points that are invariant to scale and orientation.

2. *Keypoint localization* At each candidate location, a detailed model is fit to determine location and scale. Keypoints are selected based on measures of their stability.
3. *Orientation assignment* One or more orientations are assigned to each keypoint location based on local image gradient directions. All future operations such as the keypoint description are performed on image data that has been transformed relative to the assigned orientation, scale, and location for each feature, thereby providing invariance to these transformations.

Lowé based most of his ideas on the studies of Tony Lindeberg. In 1994 Lindeberg [80] and Koenderink [75] showed that the most efficient way to implement a scale-space kernel is the use of the Gaussian function. Moreover Lindeberg showed in [80] that a Laplacian of Gaussian filter can be approximated by a Difference Of Gaussian filter, computationally less expensive.

For these reasons Lowé decided to use the Difference Of Gaussian filter for the first stage of his SIFT algorithm.

The scale space of an image is defined as a function, $L(x, y, \sigma)$, that is produced from the convolution of a variable-scale Gaussian, $G(x, y, \sigma)$, with an input image, $I(x, y)$:

$$L(x, y, \sigma) = G(x, y, \sigma)I(x, y) \quad (3.11)$$

where $*$ is the convolution operation and:

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}} \quad (3.12)$$

The difference-of-Gaussian scale-space $D(x, y, \sigma)$ can be computed from the difference of two nearby scales separated by a constant multiplicative factor k :

$$\begin{aligned} D(x, y, \sigma) &= (G(x, y, k\sigma) - G(x, y, \sigma))I(x, y) \\ &= L(x, y, k\sigma) - L(x, y, \sigma) \end{aligned} \quad (3.13)$$

Figure 3.4 shows the approach used to construct the scale-space. The original image is incrementally convolved with Gaussian kernels to produce images separated by a constant factor k in scale space, shown stacked in the left column. Adjacent image scales are subtracted to produce the difference-of-Gaussian images shown in the right. Once

a complete octave has been processed, the Gaussian image on top is subsampled by a 2 factor and smoothed incrementally like the previous octave. The accuracy of sampling remains the same, while computational costs are reduced.

In order to detect the local maxima and minima of $D(x, y, \sigma)$, each sample point is

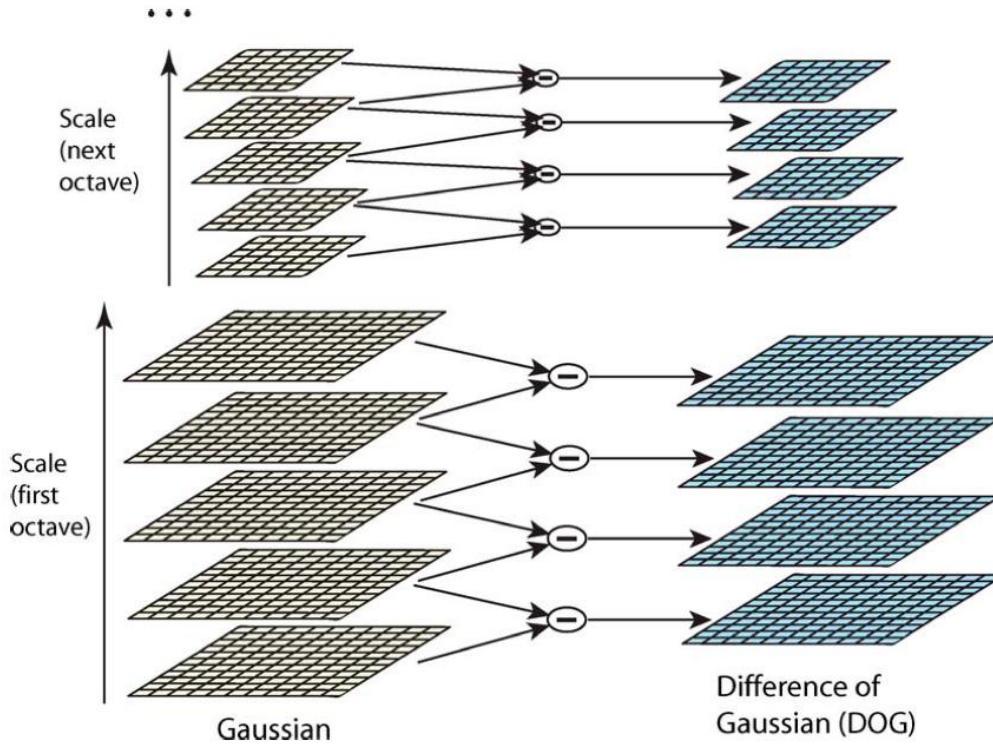


Fig. 3.4 Stack of images convolved with Gaussians on the left. Images obtained by DoG on the right. Image from [83].

compared to its eight neighbors in the current image and nine neighbors in the scale above and below.

Then it is performed an accurate keypoint localization by finding the extremum (i.e. setting derivative to zero) using the Taylor expansion to approximate the image function.

The next step is the elimination of edge responses which, as seen in the previous paragraph, have in general low repeatability and low distinctiveness. This is done by calculating the response of the Hessian filter. If the ratio between the eigenvalues is high we are in the presence of an edge. This keypoints are discarded.

The last step of the SIFT detection pipeline is the assignment of an orientation to each keypoint using a set of precomputed pixel differences. All computation are performed

Keypoint-based Approaches

in a scale-invariant way:

$$\theta(x, y) = \tan^{-1}((L(x, y + 1) - L(x, y - 1))/(L(x + 1, y) - L(x - 1, y))) \quad (3.14)$$

where $L(x, y)$ is an image sample at a given scale.

3.1.5 SURF detector

In this section is presented the Fast-Hessian detector, developed by Herbert Bay, Tinne Tuytelaars and Luc Van Gool [15] as part of the Speeded Up Robust Features (SURF) algorithm.

The main idea of the Fast-Hessian detector is the use of differences of squared boxes to approximate the second order Gaussian derivatives. This process is highly speeded up by the use of Integral Images [141] developed by Viola and Jones which permit to compute the definite integral of a given image patch using only four floating-point operations independently of the patches' size.

This consideration brings another important contribution to the speed of this detector. While a traditional scale-space is usually build as an image pyramid in witch images are repeatedly smoothed with a Gaussian and subsequently sub-sampled, the Fast-Hessian doesn't need to compute such scale-space. It just needs the up-scaling of the filter size. Here is the typical Hessian filter formula. Given a point $\mathbf{x} = (x, y)$ of an image I , the Hessian matrix \mathcal{H} in \mathbf{x} at scale σ is defined as follows:

$$\mathcal{H}(\mathbf{x}, \sigma) = \begin{bmatrix} L_{xx}(\mathbf{x}, \sigma) & L_{xy}(\mathbf{x}, \sigma) \\ L_{xy}(\mathbf{x}, \sigma) & L_{yy}(\mathbf{x}, \sigma) \end{bmatrix} \quad (3.15)$$

where $L_{xx}(\mathbf{x}, \sigma)$ is the convolution of the Gaussian second order derivative with the image I in point \mathbf{x} , and similarly for L_{xy} and L_{yy} .

Another choice made by the SURF authors to speed up the detection is the use of the Hessian cornerness measure for selecting both scale and location of a maximum. This is done despite a loss in accuracy, see Section 3.1.3.

Figure 3.5 shows Gaussian second order partial derivatives and the related squared approximations used by SURF.

3.1.6 FAST inspired detectors

FAST is the acronym of Features from Accelerated Segment Test. It has been proposed by Edward Rosten and Tom Drummond in [118]. The idea is to compare the intensity

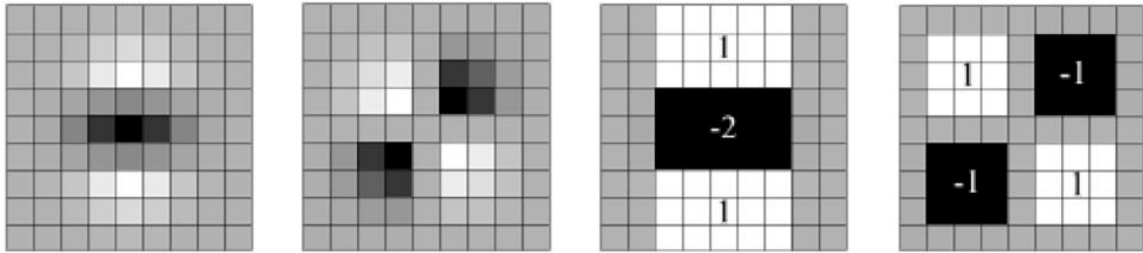


Fig. 3.5 Left to right: The (discretized and cropped) Gaussian second order partial derivatives in y -direction and xy -direction, and the approximations thereof using box filters. The grey regions are equal to zero. Images from [15].

of a single pixel with respect to a circular neighborhood to localize corners.

In the case of FAST, not the whole area of the circle is evaluated, but only the pixels on the discretized circle describing the segment. Using a test mask of diameter 3.4 pixels, 16 pixels have to be compared to the value of nucleus. To prevent this extensive test, the corner criterion has been relaxed. The criteria for a pixel to be a corner according to the accelerated segment test (AST) is as follows: there must be at least S connected pixels on the circle which are brighter or darker than a threshold determined by the center pixel value. The values of the other $16S$ pixels are disregarded. Therefore, the value S defines the maximum angle of the detected corner. Keeping S as large as possible, while still suppressing edges (where $S = 8$), increases the repeatability of the corner detector.

Figure 3.6 shows an example of corner detection with a segment of 16 pixels. With a threshold of $S = 8$ the nucleus point p is considered a corner.

It is clear that the critical choice to attain FAST speed-up is about the order of comparisons of neighbor pixels. FAST's authors, used a machine learning approach to calculate the most efficient comparison order, therefore this approach needs a training phase.

In 2010 Mair et al. developed a more efficient AGAST detector [88]. This detector is based on the same underlying ideas of the FAST detector but it uses a more detailed configuration space in order to provide a more efficient solution. Moreover the AGAST algorithm uses a training algorithm to compute the decision tree for the pixels comparisons similar to the backward induction method.

It then automatically adapts to the area which is currently processed changing the used decision tree dynamically. The idea is to build, e.g., two trees and to specialize one for homogeneous and one for structured regions. At the end of each decision path, where the corner criterion is met or cannot be fulfilled anymore, a jump to the appropriate specialized tree is performed based on the pixel configuration of this leaf. This switch

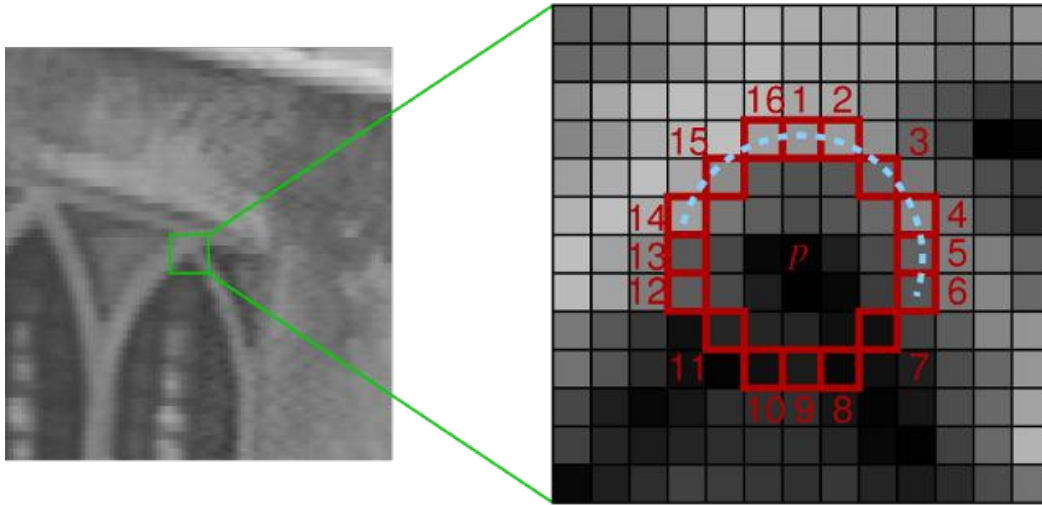


Fig. 3.6 Figure shows a 16 pixel corner detection. Images taken from [118].

between the specialized decision trees comes with no additional costs, because the evaluation of the leaf node is done offline when generating the specialized tree. In this way the AST is adapted to each image section dynamically and its performance is increased, for an arbitrary scene. Any learning becomes needless.

In 2011 Leutenegger et al. developed a new detector/descriptor called BRISK: Binary Robust Invariant Scalable Keypoints [79]. The BRISK detector is essentially a multiscale AGAST.

3.1.7 Kaze

Kaze is a detector developed in 2012 by Alcantarilla et al. [9]. Its strength relies on the scale space upon which it is built. While all the others detectors are based on a linear scale-space, Kaze uses a non-linear one. Linear scale-spaces are built with incremental blurring by a Gaussian function or an approximation of it, e.g. Difference of Gaussians or Fast Hessian. For detailed explanation see Section 3.1.4 or Section 3.1.3. This is done because linear functions for construction of scale-space are simpler and thus computationally cheaper than non-linear one. The drawback of using Gaussian blurring is that every detail is smoothed indiscriminately.

The idea of Alcantarilla et al. in [9] is the use a computational efficient approximation of Perona and Malik Diffusion Equation [109] for the scale-space building. Using this function for the scale-space construction, the blurring can be locally adapted to the structure of the image. Edges are preserved or even sharpened, while blobs are blurred, eliminating gaussian noise, jpeg artifacts etc.

The main idea is to describe the evolution of the luminance of an image as the divergence of a *flow* function that controls the diffusion process. Equation 3.16 describes the nonlinear diffusion:

$$\frac{\partial L}{\partial t} = \text{div}(c(x, y, t) \cdot \nabla L) \quad (3.16)$$

where *div* is the divergence operator and ∇ represents the gradient. c is a function of the x, y parameters representing the spatial coordinates and t , is the time of diffusion, that in our case it is to be interpreted like the scale parameter. Thus, c is a function that tends to produce a simpler representation of the original image for large values of t .

Perona and Malik in 1990 [109] proposed a c function dependent on the gradient magnitude $|\nabla L|$:

$$c(x, y, t) = g(|\nabla L(x, y, t)|) \quad (3.17)$$

in particular they made two formulations of the g function:

$$g_1 = \exp\left(-\frac{|\nabla L_\sigma|^2}{k^2}\right) \quad (3.18)$$

$$g_2 = \frac{1}{1 + \frac{|\nabla L_\sigma|^2}{k^2}} \quad (3.19)$$

Later, it was proposed By Weickert [146] another function of diffusion:

$$g_3 = \begin{cases} 1 & |\nabla L_\sigma|^2 = 0 \\ 1 - \exp\left(-\frac{3.315}{\left(\frac{|\nabla L_\sigma|}{k}\right)^8}\right) & |\nabla L_\sigma|^2 > 0 \end{cases} \quad (3.20)$$

In all these equations k is the contrast factor that controls the level of diffusion. The three function are slightly different, they have different characteristics even though they do the same thing. The function g_1 promotes high-contrast edges, g_2 promotes wide regions over smaller ones and g_3 perform strong smoothing on both sides of an edge and weak smoothing across it.

The processing pipeline is built in this way. First the parameter k estimated for every image before the execution of the real processing. Then it is built the scale-space accordingly with the diffusion equations at various values of t . Maxima in scale space are detected with a scale-normalized Hessian operator:

$$L_{Hessian} = \sigma^2(L_{xx}L_{yy} - L_{xy}^2) \quad (3.21)$$

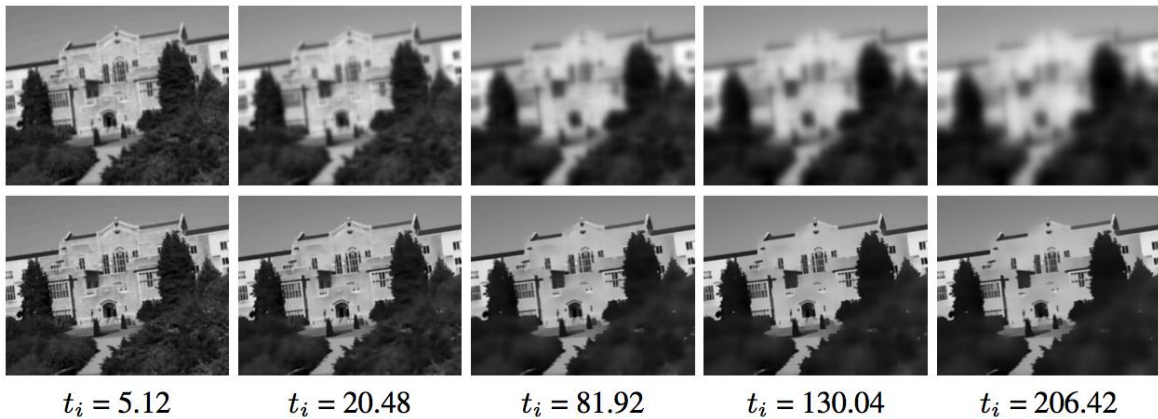


Fig. 3.7 Example of non-linear scale space construction (lower images function g_3) vs. linear scale space (higher ones). Images from [9].

with L_{xx} , L_{yy} , L_{xy} the second order derivatives. The search of extrema is performed at a coarse level, then sub-pixel refinement is done in the same way SIFT do. The dominant orientation of the keypoint is found with a method similar to the one used by the SURF detector. In the Descriptors chapter, Kaze it is not mentioned because its descriptor is almost equal to SURF.

3.2 Keypoint-based Local Descriptors

Desirable properties of descriptors are:

- *Distinctiveness*: a good descriptor must produce maximally distinctive descriptions. Image patches with different visual characteristics must correspond to distant descriptors.
- *Invariance*: a descriptor must achieve invariance to some kind of transformations, see Section 2, depending on the specific field of application.
- *Computational complexity*: computationally cheap and fast detectors are preferred. Especially for some kind of applications: robotics, real-time tracking, embedded devices etc.

In the following sections, an overview of the state-of-the-art well known descriptors is presented.

3.2.1 SIFT

The SIFT descriptor developed by Lowe [83] is based upon a model of biological vision of complex neurons in primary visual cortex. These complex neurons respond to a gradient at a particular orientation and spatial frequency, but the location of the gradient on the retina is allowed to shift over a small receptive field rather than being precisely localized.

In the same way, SIFT descriptor samples magnitude gradients on a grid superimposed over the keypoint center. To achieve invariance to rotation and scale, this grid is rotated relative to the keypoint orientation found by the detector and the sampling distance adjusted with respect to the characteristic scale of the keypoint.

A gaussian function of σ equal to an half of the grid diameter lying upon the grid is used for weighting the contribution of each gradient. This is done to give less emphasis to gradients that are far from the center of the keypoint, as they are most affected by misregistration errors.

The grid is also split into 4x4 cells, than the weighted contribution of each gradient lying inside a cell is merged into a single histogram. This histogram collects gradient magnitudes over 8 discretized directions. Thus the typical SIFT descriptor dimension is 128 because of the concatenation of 4x4 8-bins histograms. Figure 3.8 shows the sampling grid and the 4x4 cell division.

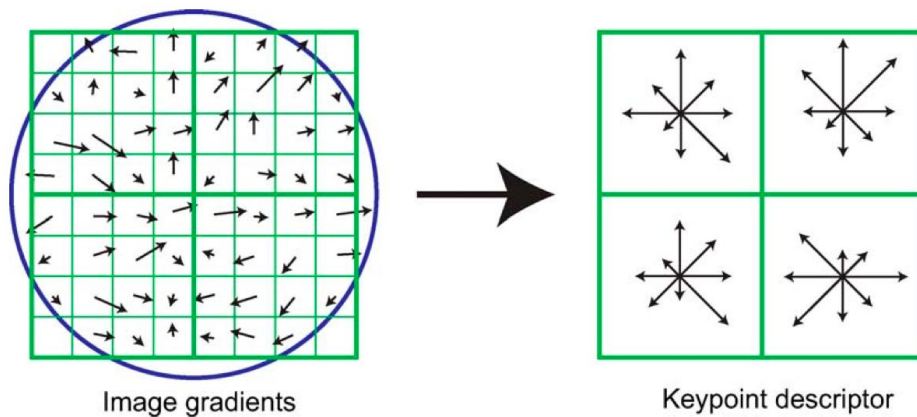


Fig. 3.8 SIFT descriptor sampling structure. Image from [83]. On the left is depicted the sampling pattern grid. The blue circle represents the gaussian smooth, even though, of course, the weights decrease smoothly. On the right is shown the 4x4 division and the 8-bins histograms.

3.2.2 SURF

The SURF descriptor [15] is based on similar principles underlying the SIFT descriptor, but with some changes to improve description speed. Like in SIFT the squared sampling region is rotated with respect to the keypoint angle, scaled and splitted up into smaller 4x4 regions to maintain spatial information.

Unlike SIFT, the SURF descriptor does not sample gradients magnitudes. It makes use of the Haar wavelet filter and sample every response in vertical and horizontal direction. For every region, it is summed up every response in the two directions and even the sum of absolute value. Therefore, if we call d_x the horizontal response and d_y the vertical response the final descriptor is made by concatenation of 4x4 vectors $\mathbf{v} = (\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|)$. Thus the descriptor length is 64.

Figure 3.9 shows the filter response for three different kinds of regions demonstrating high distinctiveness.

Authors made available an extended version of this descriptor which descriptors

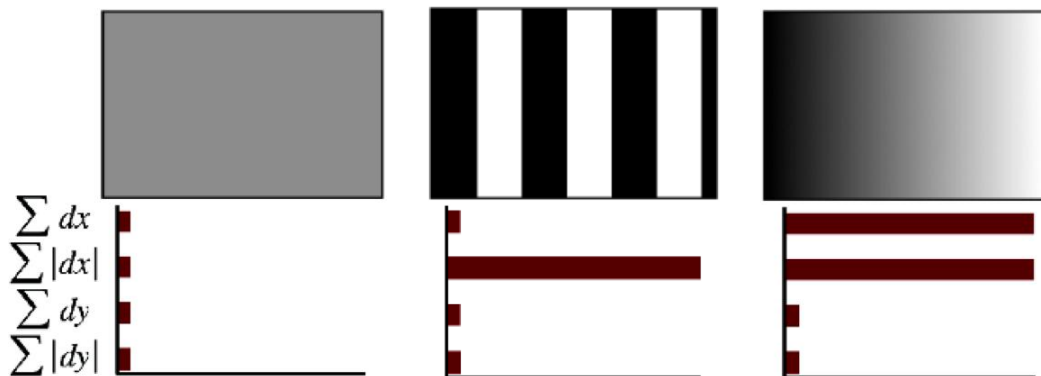


Fig. 3.9 SURF descriptor structure: histograms of Haar wavelet horizontal and vertical responses. In addition, histograms made of sums of absolute values. Results obtained from three different regions. Figure from [15].

produced are 128-dimensional. This version is obtained by computing the sums of d_x and $|d_x|$ separately for $d_y < 0$ and $d_y \geq 0$ and the same with d_y and $|d_y|$. They have made even an up-right version suited for applications where the camera remains more or less vertical, this variant skip the keypoint orientation assignment stage.

3.2.3 FREAK

The FREAK descriptor, acronym for Fast Retina Keypoint, was developed in 2012 by Alahi et al. at the EPFL of Lausanne [7]. It is built upon the same detector module as BRISK [79], i.e. a multiscale AGAST [88] detector (see Section 3.1.6). It is a descriptor

3.2 Keypoint-based Local Descriptors

module inspired to mechanisms behind the human retina photoreceptors.

Inside the retina lie several photoreceptors. Every photoreceptor is wired with a few others to a ganglion cell. The region where light influences the response of a ganglion cell is the receptive field. Its size increases with the radial distance from the foveola, i.e. center of the retina.

In the same way the FREAK descriptor use Gaussian kernels of different sizes with respect to the log-polar retinal pattern. In addition, the sampling patterns are overlapped to add redundancy and therefore discriminative power. Figure 3.10 shows this configuration.

FREAK, unlike the previous algorithms, produces binary descriptors, i.e. descriptors

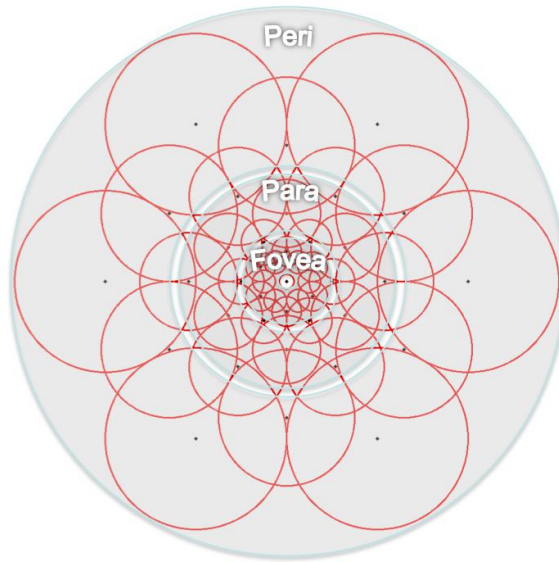


Fig. 3.10 FREAK sampling pattern similar to the retinal ganglion cells distribution with their corresponding receptive fields. Each circle represents a receptive field where the image is smoothed with its corresponding Gaussian kernel. Figure from [7]

built by concatenation of bits.

Every descriptor is built by thresholding the difference between pairs of receptive fields with their corresponding Gaussian kernel:

$$F = \sum_{0 \leq a < N} 2^a T(P_a) \quad (3.22)$$

Keypoint-based Approaches

with F the final descriptor, N is the size of descriptor, $T(P_a)$ is a pair of receptive fields and:

$$T(P_a) = \begin{cases} 1 & \text{if } I(P_a^{r_1}) - I(P_a^{r_2}) > 0, \\ 0 & \text{otherwise,} \end{cases} \quad (3.23)$$

with $I(P_a^{r_1})$ is the smoothed intensity of the first receptive field of the pair P_a .

FREAK's authors made various experiments and discovered that the best compromise between distinctiveness and performances brings to a descriptor of size 512. They also made a selection based on the minimization of entropy of the most significant receptors pairs against all possible pairs.

Alahi et al. proposed also a matching technique which mimic the saccadic search of the human eye. On every descriptor is performed a cascade of comparisons accelerating the matching process. Initially, only the first 16 bit are compared, then, if the difference is under a threshold the matching process can go further and comparing the last bits.

3.2.4 Color descriptors

Descriptors exposed in previous sections make use of luminance only and thus ignore color information. Depending on the type of objects to recognize and the application, color can be a very discriminative clue. In Figure 3.11 is shown an example of scene involving objects identical to each others except for their color. A grayscale descriptor



Fig. 3.11 Image representing a can of tea lemon-flavoured surrounded by cans of tea peach-flavoured. Luminance based descriptors could not find the lemon one among others.

will probably fail because is totally blind to color differences.

Nevertheless, for this kind of descriptors it is more difficult to achieve invariance to

3.2 Keypoint-based Local Descriptors

illumination conditions, because they are not, by definition, invariant to color light changes. For an in depth review of common light transformations see Section 2.1.3).

Color descriptors can be classified in two classes depending on the approach used to combine the shape, or *luminance*, and color information [134][71][122]. Some algorithms make use of an Early Fusion approach and others adopt a Late Fusion approach. As defined by Khan et al. in [122]: “Early fusion combines shape and color at the pixel level, which are then processed together throughout the rest of the description pipeline. In late fusion, shape and color are described separately from the beginning and the exact binding between the two features is lost.”

Basically, every Early Fusion description pipeline considered consists of the following steps:

1. Transformation of the image channels into a specific color space.
2. Computation of SIFT descriptor on each color space channel.
3. Concatenation of descriptions computed over each channel.

Specifically RGB SIFT [134] computes SIFT descriptors on the original red, green, and blue channels of the image and then concatenates them, thus keeping the image in its original color space. Opponent SIFT [134] instead applies the following transformation from RGB to opponent color space $O_1O_2O_3$:

$$\begin{pmatrix} O_1 \\ O_2 \\ O_3 \end{pmatrix} = \begin{pmatrix} \frac{R-G}{\sqrt{2}} \\ \frac{R+G-2B}{\sqrt{6}} \\ \frac{R+G+B}{\sqrt{3}} \end{pmatrix} \quad (3.24)$$

Transformed Color SIFT [134] normalizes the RGB channels independently into zero-mean and unity-variance $R'G'B'$ channels:

$$\begin{pmatrix} R' \\ G' \\ B' \end{pmatrix} = \begin{pmatrix} \frac{R-\mu_R}{\sigma_R} \\ \frac{G-\mu_G}{\sigma_G} \\ \frac{B-\mu_B}{\sigma_B} \end{pmatrix} \quad (3.25)$$

where μ_C is the mean and σ_C the standard deviation of the distribution in channel $C = \{R, G, B\}$. HSV SIFT [134] computes SIFT descriptors over all three channels of

Keypoint-based Approaches

the HSV color model:

$$\begin{pmatrix} H \\ S \\ V \end{pmatrix} = \begin{pmatrix} \text{atan2}(\beta, \alpha) \\ 0 \text{ if } V = 0, \frac{\sqrt{\alpha^2 + \beta^2}}{V} \text{ otherwise} \\ \max(R, G, B) \end{pmatrix} \quad (3.26)$$

where $\alpha = (2R - G - B)/2$ and $\beta = \sqrt{3}(G - B)/2$. C-SIFT [134][6] applies the C -invariant [49] to the O_1 and O_2 channels of the opponent color space to eliminate the remaining intensity information from these channels. This can be intuitively seen as the normalized opponent color space O_1/O_3 and O_2/O_3 . The rg SIFT transforms the image in the normalized RGB color model, where the chromaticity components r and g describe the color information (b is omitted since it is redundant as $r + g + b = 1$):

$$\begin{pmatrix} r \\ g \\ b \end{pmatrix} = \begin{pmatrix} \frac{R}{R+G+B} \\ \frac{G}{R+G+B} \\ \frac{B}{R+G+B} \end{pmatrix} \quad (3.27)$$

The last Early Fusion color descriptor considered, i.e. oRGB SIFT [140], maps the image into oRGB color space, which is an opponent color space that is ideal for RGB computation [28]. The mapping consists in two steps: the first one is a linear transformation from RGB to LC_1C_2 :

$$\begin{pmatrix} L \\ C_1 \\ C_2 \end{pmatrix} = \begin{pmatrix} 0.299 & 0.587 & 0.114 \\ 0.500 & 0.500 & -1.000 \\ 0.866 & -0.866 & 0.000 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix} \quad (3.28)$$

The second one is the transformation from LC_1C_2 to oRGB, which consists in a compression or decompression of angles depending on which quadrant the linearly transformed point ends up in [28].

Conversely, every Late Fusion description pipeline extracts the shape information from the gray-level image. Then, the color descriptor is computed directly from the original image. Finally, the shape and color descriptions are merged as follows:

1. Normalization of the two parts separately (color and shape descriptions).
2. Multiplication by a fusion factor depending on the specific descriptor.
3. Concatenation (of the color and shape descriptors).
4. Normalization of the overall description.

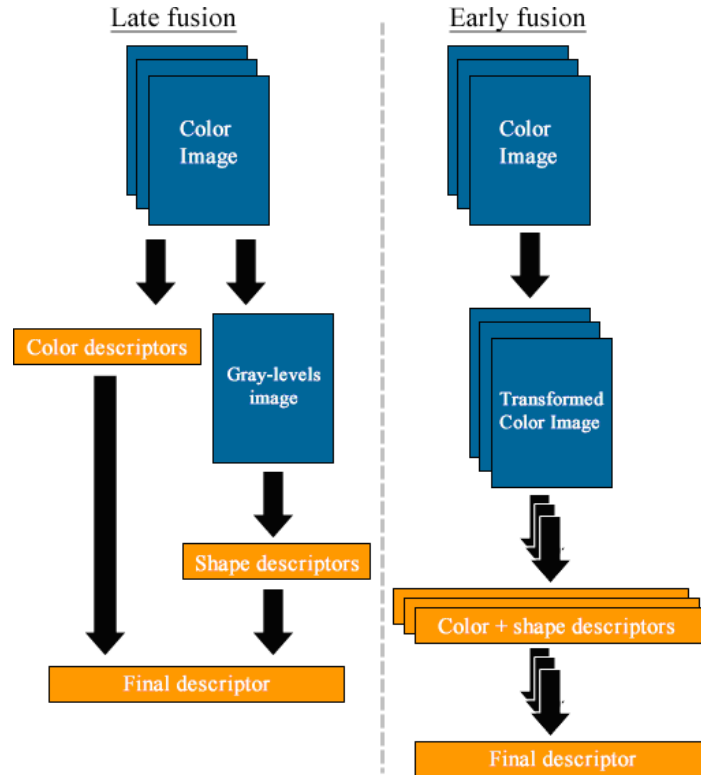


Fig. 3.12 Late Fusion vs Early Fusion. The Late Fusion approach computes the color descriptors from the original image and the shape descriptors from a gray-levels image; while the Early Fusion approach computes all the descriptors from every color channel (shape and color information are correlated) and then merges them.

Fig. 3.12 depicts the pipelines of the Early and Late Fusion approaches.

All the Late Fusion descriptors, with the exception of the Hue SIFT algorithm [134], require a prior training phase which determines a quantization function to map each RGB pixel value into a probability vector defining the likelihood of an RGB pixel value to represent a certain color. Every algorithm is characterized by a unique and specific methodology to build the map function. Hue SIFT introduces a concatenation of the robustified hue histogram with the SIFT descriptor. The hue histogram is made more robust by weighing each sample of the hue by its saturation, since the certainty of the hue is inversely proportional to the saturation. Color Names descriptor [138] is trained from weakly labeled images returned by Google Image search. Fuzzy Sets Color Names function [17] trains by using parametric membership functions defined on the basis of psychophysical data obtained from a color-naming experiment. Both the methods based on Color Names are inspired from [122]. Each keypoint is described with a gray-scale SIFT concatenated with the Color Names descriptor computed as follows: the image patch centered on the keypoint (whose width depends on the keypoint scale)

Keypoint-based Approaches

is scaled to a fixed size, and for each pixel the Color Name descriptor is computed. The descriptors are weighted by a Gaussian peaked on the center of the resized patch, and then normalized. The Discriminative Color Descriptor [72] performs its training phase by using a statistical method based on information theory: it learns color descriptors which have optimal discriminative power for a specific classification problem. The problem of learning a color descriptor is solved by finding a partition of the color space using the Divisive Information-Theoretic Clustering (DITC) [38].

The computational complexity of color descriptors strongly depends on the fusion approach adopted. Every Early Fusion algorithm reveals a complexity about three times higher than that of the SIFT descriptor computed on the same gray-levels image (except for the rg-SIFT which is about two times higher). The computational complexity of every Late Fusion algorithm consists of the sum of the complexities of the following steps:

1. SIFT algorithm (computed on the gray-levels image).
2. Pyramid construction.
3. Color quantization by means of a lookup-table (with the complexity depending on the number of keypoints and their size).

3.3 Experimental Setup and Datasets

In this Section, first it is introduced the MPEG CDVS Test Model Framework, a standardized test environment for instance object recognition. Then the type of tests performed within the Test Model and the datasets are discussed. A new *Supermarket Milan* dataset is introduced for a new use case. Finally the results of a quantitative analysis of the detectors and descriptors presented in the previous Sections is exposed.

3.3.1 The MPEG CDVS Test Model Framework

MPEG CDVS [105] is a technology in the last phase of the ISO standardization process that will enable the design of efficient and interoperable visual search applications and in particular the development of technologies for visual content matching from still images. Visual content matching includes matching of views of objects, landmarks, and printed documents that is robust to partial occlusions as well as changes in vantage point, camera parameters, and lighting conditions. It has the goal of defining a standard bitstream, which encodes in compressed form the information required to perform a

search on the server’s side. The information encoded consists of a global descriptor, which is a digest computed from compact descriptors features extracted from the image, a compressed descriptor and the associated coordinates.

The CDVS Test Model (TM) [105] implements the required functionality for the extraction and comparison of compact descriptors constrained to a set of predetermined descriptor lengths.

In particular, two procedures for descriptor comparison are implemented in TM, aiming at reproducing two fundamental tasks for real visual search systems: pairwise matching and retrieval. The former regards automated verification of whether two images depict the same objects or scene; in this case, descriptors extracted from a query image are matched against the descriptors of a reference image, in order to determine whether they match or not. The latter regards the search and discovery of images contained within a large collection that depict the same objects or scenes as those depicted by a query image; this requires the database images to be processed for the creation of a database which may be searched using the descriptors extracted from the query.

The Pairwise Matching Stage compares the query and reference image descriptors to determine if the images depict the same object or scene. It uses first local descriptor matching and if the score is below a threshold, it performs global descriptor matching. If the final score is greater than a threshold, the images likely depict the same objects (a match), otherwise the object are different (a non-match).

The Retrieval Stage searches and retrieves relevant images, belonging to a large collection, that depict the same object or scene represented in the query image. At first, an off-line step processes the collection to create a database of local and global visual descriptors which can be matched against the descriptors extracted on the fly from the query. The retrieval stage performs a search in two steps, first using global descriptor to select a shortlist of matching images and then using the shortlist in the next step to compare encoded local descriptor using the Hamming distance. The final ranking score and inlier selection is computed by a geometric consistency check performed to determine the inliers among the interest point matches for the two images. The TM uses the histogram of logarithmic distance ratios (LDR) [131].

3.3.2 Pairwise Matching Experiment

A Pairwise Matching Experiment has been performed on different datasets in the framework of the CDVS Test Model [5] (See also Section 3.3.1). In this experiment, a reference image has been compared with an image describing the same object (or

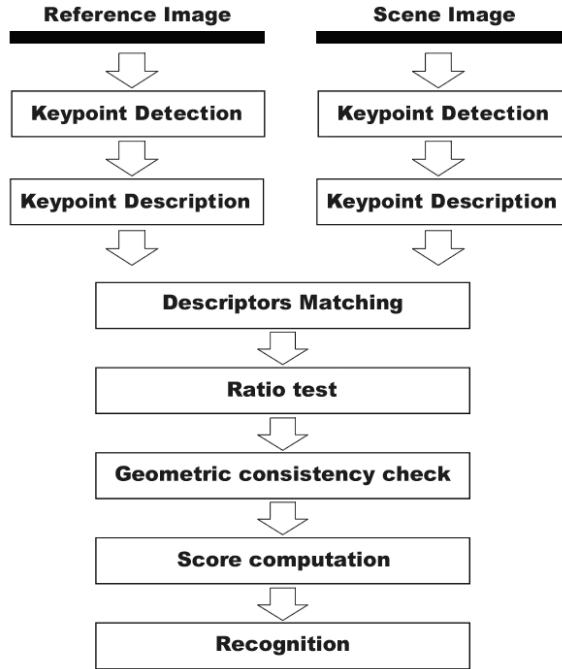


Fig. 3.13 Pairwise matching experiment evaluation Pipeline

scene) taken under different illumination conditions, with different acquisition devices, and from different points of view. Fig. 3.13 shows the complete evaluation pipeline.

Descriptors matching was accomplished by Euclidean distance for integer descriptors (i.e. SIFT, SURF, KAZE, A-KAZE) and by Hamming distance for binary descriptors (i.e. FREAK). The ratio test on candidate matches was the same proposed by Lowe [85]. Let a_1 be a descriptor from image A , b_1 and b_2 the most and second most similar descriptors from image B , and $dist(\cdot, \cdot)$ the distance function between two descriptors. If:

$$\frac{dist(a_1, b_1)}{dist(a_1, b_2)} < threshold \quad (3.29)$$

then the candidate match (a_1, b_1) was accepted as the best match candidate, otherwise it was rejected. The value of threshold was empirically set. All the accepted matches were further evaluated by a geometric consistency check. This was done using the Logarithmic Distance Ratio [131] algorithm.

The image pair scores were computed using all the geometric consistent matches. First, an image correspondence score based on all matches was computed as follows:

$$w = \sum_{\text{all matches}} \left(\cos \left(\frac{\pi}{2} \sqrt{\frac{dist_1}{dist_2}} \right) \right) \quad (3.30)$$

3.3 Experimental Setup and Datasets

Dataset	MP	NMP	Origin
1. CDs, DVDs, books, business cards (Mixed text + graphics)	3000	29903	Stanford Mobile Visual Search
2. Museum paintings	363	3639	Stanford MVS
3. Video frames	399	3999	Stanford MVS
4. Landmarks and buildings (Zurich, Turin)	1789	17948	ZuBud, Telecom Italia
5. Common object or scenes	2549	21307	University of Kentucky

Table 3.1 Datasets adopted in the MPEG CDVS test model

where $dist_1$ and $dist_2$ are the best distance and the second best distance for each match. Then, the image correspondence score was transformed by mean of the following equation to obtain the final Image Pair Score (IPS):

$$IPS = \frac{w}{w + wm_{Thresh}} \quad (3.31)$$

where wm_{Thresh} is a threshold that was empirically found for each detector/descriptor couple. Image pairs with $IPS > 0.5$ were considered a match.

All images were resized to VGA resolution (i.e. minor and major axis length equal to 480 and 640 pixels respectively). Results were reported in terms of True Positives Rate (TPR) at a given level of False Positives Rate (FPR) (also called False Accept Rate or FAR).

3.3.3 Datasets

For the gray-level descriptors comparison, the experiments were performed on five datasets of different objects, whereas for the color descriptors, the experiments were performed on the five datasets plus a recent additional dataset that we entitled “SuperMarket Milan”.

The CDVS standard datasets are listed in Table 3.1, together with the number of Matching Pairs (MP) and Non-Matching Pairs (NMP) provided as ground-truth.

The “Mixed-text + Graphics”, “Museum paintings”, and “Video frames” datasets were collected by the Stanford Mobile Visual Search research group. “Mixed-text + Graphics” consists of 2500 images of five different categories of objects: CDs, DVDs, books, text documents and business cards. The “Video frames” dataset contains 500 images captured by a mobile phone camera shooting a TV screen. Pictures in these three datasets have been shot with different cameras and various different lighting conditions, rotations, scales, and viewpoints. Fig. 3.14 depicts an object from the “Mixed-text + Graphics” dataset.

Keypoint-based Approaches

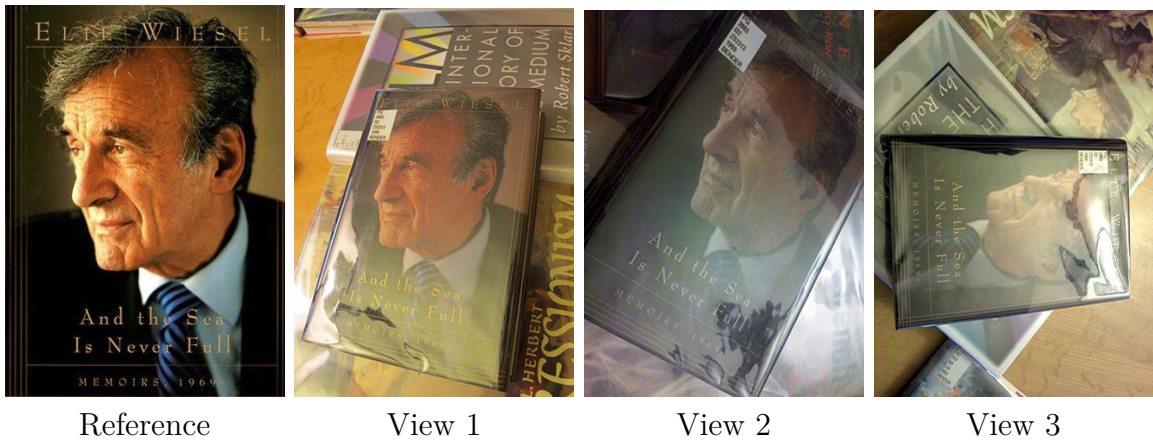


Fig. 3.14 Images of a typical object (DVD) from the “Mixed-text + Graphics” dataset.



Fig. 3.15 Images of the same building from different views from the “Turin” dataset.

The “Landmarks and Buildings” dataset includes images from two different origins: “Zurich Buildings” dataset and “Turin” dataset. “Zurich Buildings” contains pictures of 200 buildings in Zurich, shot by two cameras under different viewing conditions whereas dataset “Turin” contains images of 180 landmarks/buildings in Turin, Italy. 1440 images are still images and 540 images were extracted from videos. Pictures in this datasets exhibit different point of views of the same landmark/building, occlusions and strong light changes. Example images taken from the “Turin” dataset are depicted in Fig. 3.15.

The fifth dataset depicts “Common objects or Scenes” collected by University of Kentucky. It contains 10200 images of 2550 different objects/scenes, each shot from 4 different views. Typical image transformations include rotation, different scales, different points of view and slight changes in illumination. Fig. 3.16 shows different shots of a typical object from this dataset.

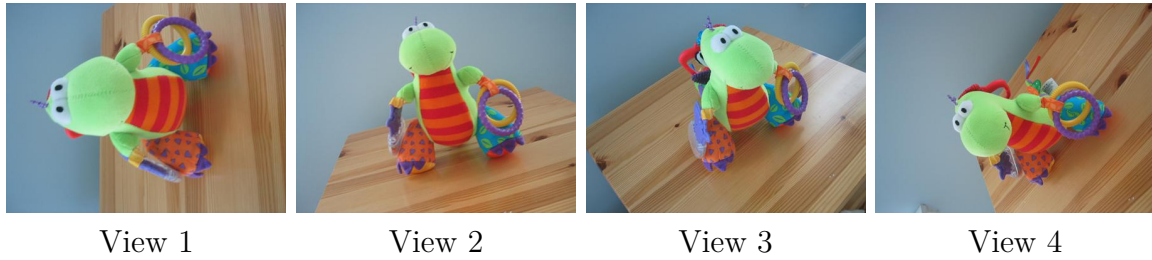


Fig. 3.16 Images of the same object from different views from the “Common objects and scenes” dataset by University of Kentucky.

3.3.4 SuperMarket Milan Dataset

“SuperMarket Milan” is a recent ad-hoc dataset composed of 1,686 photos of various supermarket products. This dataset was specifically created to test the CDVS Test Model [5] on a new use case and to report the ability of the model to recognize goods commonly available at supermarket premises¹. Photos were taken under different illumination conditions in different types of physical locations (e.g. home and supermarket shelves). Different camera devices acquired the photos, i.e.: iPhone 4, iPhone 3, Samsung Next Turbo, Samsung S Advance, Samsung Galaxy S3, Lg Optimus L5, and LG Nexus 4. Image resolution ranges from 0.1 to 5.0 MegaPixels. An annotation file containing the coordinates of vertices of the bounding quadrilateral enclosing the object is associated to each image.

The dataset is composed of reference and query images. The reference images are photos of objects taken with the iPhone 4 in the best environmental conditions, which means, for example: no objects in the background, no portions of the object are occluded, etc. The query images are photos of the object in its context, either on the supermarket shelf or at home. Fig. 3.17 shows an example of an object. The left photo represents the reference image and the others are examples of queries.

There are 1,697 query images and 430 reference images (total 2,127). About four query images for each reference. Two different pairwise matching experiments were conducted with the “SuperMarket Milan” dataset. In the first experiment (noted as 6a in the rest of this paper) all the matching and non-matching pairs have been chosen randomly. In the second experiment (6b), the matching pairs were also chosen randomly, but the non-matching pairs were chosen randomly among the photos of so-called color significant products. In particular, this experiment measured the discriminative power of color descriptors. The phrase “color significant products” refers to products that

¹The “SuperMarket Milan” dataset was presented at the 105th MPEG Meeting (Vienna) and tested on the CDVS Test Model.



Fig. 3.17 Images of the same object from different points of view from “SuperMarket Milan” dataset.

differ from each other’s by color and not by shape. Fig. 3.18 shows examples of color significant products.

For the pairwise matching experiment 6a a total of 3350 MP and 33506 NMP are provided; 250 MP and 2522 NMP are instead provided for the experiment 6b. For the retrieval experiment 1686 queries are provided.

3.4 Quantitative Evaluation

3.4.1 Algorithms considered for comparison

In this section is presented a list of the gray-level and color algorithms evaluated. LoG or Laplacian of Gaussian and SIFT, acronym for *Scale-Invariant Feature Transform* [84][85] are the algorithms adopted, respectively, for keypoint detection and local visual description in the CDVS Test Model ver. 10 [5]. The following implementations have been considered for testing: Original Lowe’s binary code [84][85]; VLFeat [139] and the OpenCV library implementation [27]. Different affine-invariant keypoint detectors [93] have been investigated together with the VLFeat SIFT descriptor: Hessian, Hessian Laplace, Multiscale Hessian, Harris-Laplace, Multiscale Harris and Difference of Gaussians (DoG). These detectors normalize the image patch around each detected interest point according to the estimated affine transformation. They also differ in the strategy used to achieve scale-invariance: Laplace automatically selects a single characteristic scale of a keypoint, whereas Multiscale may associate multiple scales to the same keypoint.



Fig. 3.18 Examples of color significant products from the “SuperMarket Milan” dataset.

Keypoint-based Approaches

Name	Patent	Reference
SIFT detector and descriptor	Patent US 6711293 B	[85]
SURF detector and descriptor	Patent US 8165401 B	[16]
Harris detector	Free	[93]
Hessian detector	Free	[93]
FREAK descriptor	Free	[8]
KAZE detector and descriptor	Free	[10]
A-KAZE detector and descriptor	Free	[11]

Table 3.2 Gray-levels algorithms investigated

The other gray-levels algorithms that were considered are: the OpenCV SURF (Speeded-Up Robust Features) [16] implementation; the OpenCV FREAK (Fast Retina Keypoint) [8] implementation which uses SURF as the keypoint detector, and the KAZE [10] and A-KAZE [11] original code.

All tests were made with the default parameters for each algorithm with the exception of the response filter threshold. This parameter was set so each detector produced on average about 1,000 keypoints per VGA image, therefore controlling the number of interest points generated. Indeed the distinctiveness of the descriptors produced was analyzed instead of their quantity; therefore their quantity was limited to make fair comparisons. Moreover, in the CDVS processing pipeline the detected keypoints and their descriptors must be sent over the network to a server and, since the network is bandwidth constrained, it was reasonable to limit the maximum number of keypoints produced as an attempt to fit their binary representation into the network bitrate available. Table 3.2 lists the investigated gray-levels algorithms with their licenses and links to sources or binaries.

Table 3.3 lists all the color descriptors considered for comparison. For every color descriptor evaluated, the type of fusion approach and the dimension (in bytes) of the color description produced is reported. To perform fair comparisons, every color descriptor was associated with the same keypoints detector (i.e. the VLFeat implementation of SIFT). Detection phase was performed on the gray-levels image.

3.4.2 CDVS Evaluation on SuperMarket Milan Dataset

According to the MPEG CDVS Evaluation procedure protocol [4], a full characterization of the Test Model has been performed on the “SuperMarket Milan” dataset. The results were evaluated using two types of experiments: retrieval and pairwise matching. Both experiments were done on the six different datasets described in the previous

3.4 Quantitative Evaluation

Name	Dimension	Fusion	Reference
RGB SIFT	384	Early	[134]
Opponent SIFT	384	Early	[134]
Transformed Color SIFT	384	Early	[134]
HSV SIFT	384	Early	[134]
C-SIFT	384	Early	[134][6]
rg SIFT	256	Early	[134]
oRGB SIFT	384	Early	[140]
Hue SIFT	164	Late	[134]
Color Names	139	Late	[137][138]
Fuzzy Sets Color Names	139	Late	[17]
Discriminative Color	139, 153, 178	Late	[72]

Table 3.3 Evaluated color descriptors

section. Results were reported for the following operating points (upper bounds on average descriptor lengths in each experiment): 512, 1K, 2K, 4K, 8K, 16K bytes as query length.

The results for the retrieval and pairwise matching were evaluated using different sets of measures. Retrieval results were measured in terms of Mean Average Precision (MAP) and success rate for Top Match.

Pairwise matching results were measured in terms of Success Rate (i.e. TPR) at the average FPR of 1%. In fact, the CDVS Test Model is optimized to give an average FPR of 1% on the five standard CDVS datasets (i.e. dataset 1 to 5). The performances relative to the new datasets (i.e. 6a and 6b) have been obtained applying the Test Model as is. Results of the pairwise matching experiment 6a were in line with those of the other datasets. The success rate varied between 76.60% for lower bitrate descriptors and 92.78% for higher bitrate descriptors (i.e. 8k) and the range of FPR was from 0.62% for the highest bitrate descriptor to 1.35% for the lower bitrate descriptors (i.e. 512-2k). The plot of the success rate and FPR with respect to descriptor length for all the datasets considered are respectively reported in Fig. 3.19.a and Fig. 3.19.b.

For the pairwise matching experiment 6b, Success Rates values were similar to those of the experiment 6a (see Fig. 3.19.a), but FPR showed important differences. The range was from 37.83% to 65.94%. The highest level of FPR for this experiment was found in the highest bitrate descriptor, as can be seen in Fig. 3.19.b.

These levels showed a clear inability of the Test Model to discriminate “color significant products” (as previously defined). In particular, high levels of Success Rate, demonstrated the ability of the Test Model to recognize the class of object of dataset 6 (i.e. “SuperMarket Milan” dataset) but high levels of FPR also revealed that the

Keypoint-based Approaches

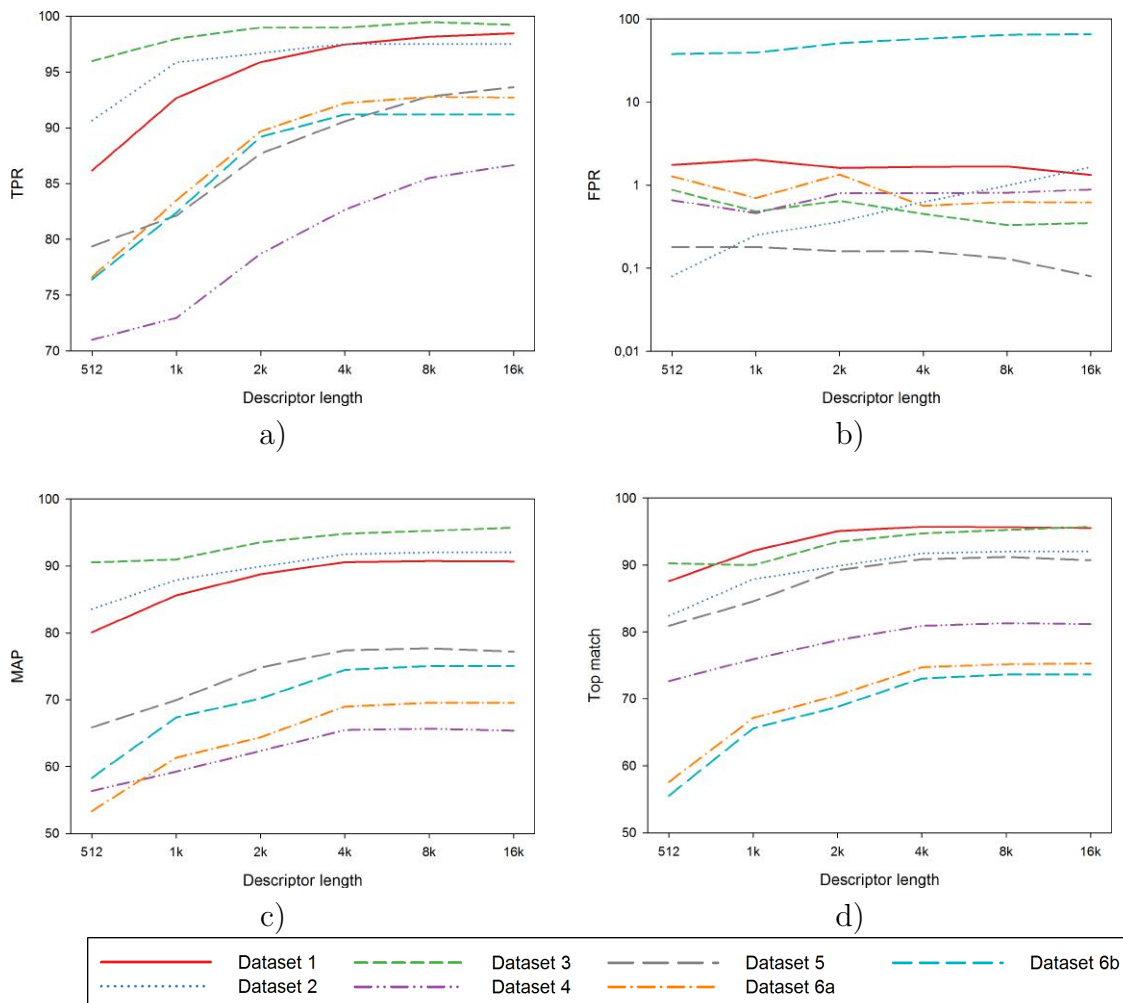


Fig. 3.19 Plots of the TPR (a), FPR (b), MAP (c) and Top Match (d) with respect to descriptor length for all the datasets considered.

current Test Model was not able to differentiate between two similar objects. As a consequence, the average FPR levels were extremely high.

On the Retrieval experiment, MAP values of datasets 6a and 6b were similar to those of the datasets 4 and 5. While datasets 1, 2 and 3 had high levels of MAP (i.e. from 76.56% to 95.78%), datasets 4 and 5 showed lower MAP levels (from 56.34% to 77.42%). The retrieval experiments 6a and 6b, concerning the “SuperMarket Milan” dataset showed levels slightly lower than those of datasets 4 and 5: from 53.34% to 75.06%. The Top Match success ratios were very low with respect to those of the others datasets. The normal range for this value was between 72.78% and 96.27% while the range of Top Match values of the retrieval experiments 6a and 6b were from 55.52% to 75.33%. As expected, the Top Match values for the experiment on dataset 6b were slightly lower than those on dataset 6a. The plot of MAP and Top Match with respect to descriptor length for all the datasets considered are respectively reported in Fig. 3.19.c and Fig. 3.19.d.

The retrieval data confirmed what could be deduced from the pairwise experiment results. Retrieval of SuperMarket Dataset objects was slightly more difficult compared to other kinds of objects of the CDVS Test Model dataset because of the shape of the supermarket objects themselves. Moreover the Top Match statistics showed that the current algorithms of the Test Model were not highly discriminative for this category of objects and in particular they were troubled by very similar “color significant products”.

Given the low performance of the CDVS Test Model on the datasets 4, 5, and 6, in the next subsections a set of gray-level and color descriptors will be tested. The experiment is aimed to see if there are alternative descriptors that can consistently improve the performance on all the datasets with respect to those obtained by the descriptor actually used in the CDVS Test Model.

3.4.3 Gray-level Descriptors

To test detectors and descriptors, two different types of experiments were conducted and reported for each dataset. In the first experiment, all the detected keypoints were used in the matching phase. This experiment aims to assess the upper bound performance of the descriptors outside the CDVS Test Model framework. In the latter experiment, a selection of the most discriminant 1,024 detected keypoints was made on the basis of the strength of the filter response before the matching phase was performed. This experiment follows the CDVS guidelines, and uses the Test Model disabling the descriptor compression step, which has to be designed ad-hoc for each different descriptor.

Keypoint-based Approaches

True Positives Rates (TPR) were measured at two different levels of FPR: 10% and 1% (as MPEG CDVS requires). The obvious expectation is that the measured TPR should not be lower for any image at the higher FPR level. This expectation was confirmed by the experiments reported in Fig. 3.20 and 3.21. Concerning gray-level detectors and descriptors, KAZE performed equal or better than SIFT on the third, fourth and fifth datasets. SURF obtained very good results on the first three datasets. FREAK performed worse than SURF on all the datasets. Performance of A-KAZE are always lower than those of KAZE.

Affine invariant detectors, as expected, worked well under viewpoint changes. Their overall results were poorer than the SIFT and KAZE algorithms for all datasets, except for the “Video Frames” dataset. Conversely, on the “Landmarks and Buildings” dataset, affine invariant detectors achieved the best performances among all the algorithms. Limiting the number of keypoints to 1024 particularly affects the performance of some descriptors: DoG, Harris Laplace, Hessian Laplace, Multiscale Harris, Multiscale Hessian, SURF-FREAK, and SURF. This is particularly evident on datasets 1 to 5 where the loss in TPR shows a magnitude up to 0.20 (SURF-FREAK on dataset 2). The loss in TPR is much lower on datasets 6a and 6b, where the magnitude is lower than 0.032.

Table 3.4 shows the average computational time required by each algorithm to detect the keypoints and extract the features. Indeed a point of focused attention was not only the accuracy aspect of these algorithms but also their complexity as a measure of the computational burden on the implementation. For each detector the filter response threshold value, the average number of keypoints detected, and the average overall time spent for detection and description were reported. Timings have been taken on the images belonging to the “Mixed text + graphics” dataset (scaled at VGA resolution) with an x86 2,4 GHz single processor with 3 MB of Cache L2 and 8 GB of RAM. As expected, A-KAZE confirms to be the algorithm showing the lowest average computational time [11]. Quite remarkably, the SIFT (OpenCV) implementation is the second among the lowest complexity methods, which is contrary to the common belief, and it was not only the most accurate on average, but also the one showing the best trade-off between performance and speed, making it the most likely candidate for embedded system mapping.

3.4.4 Color Descriptors

Comparisons of color descriptors are shown with respect to a baseline algorithm. The chosen baseline was the VLFeat implementation of SIFT (gray-levels). The

3.4 Quantitative Evaluation

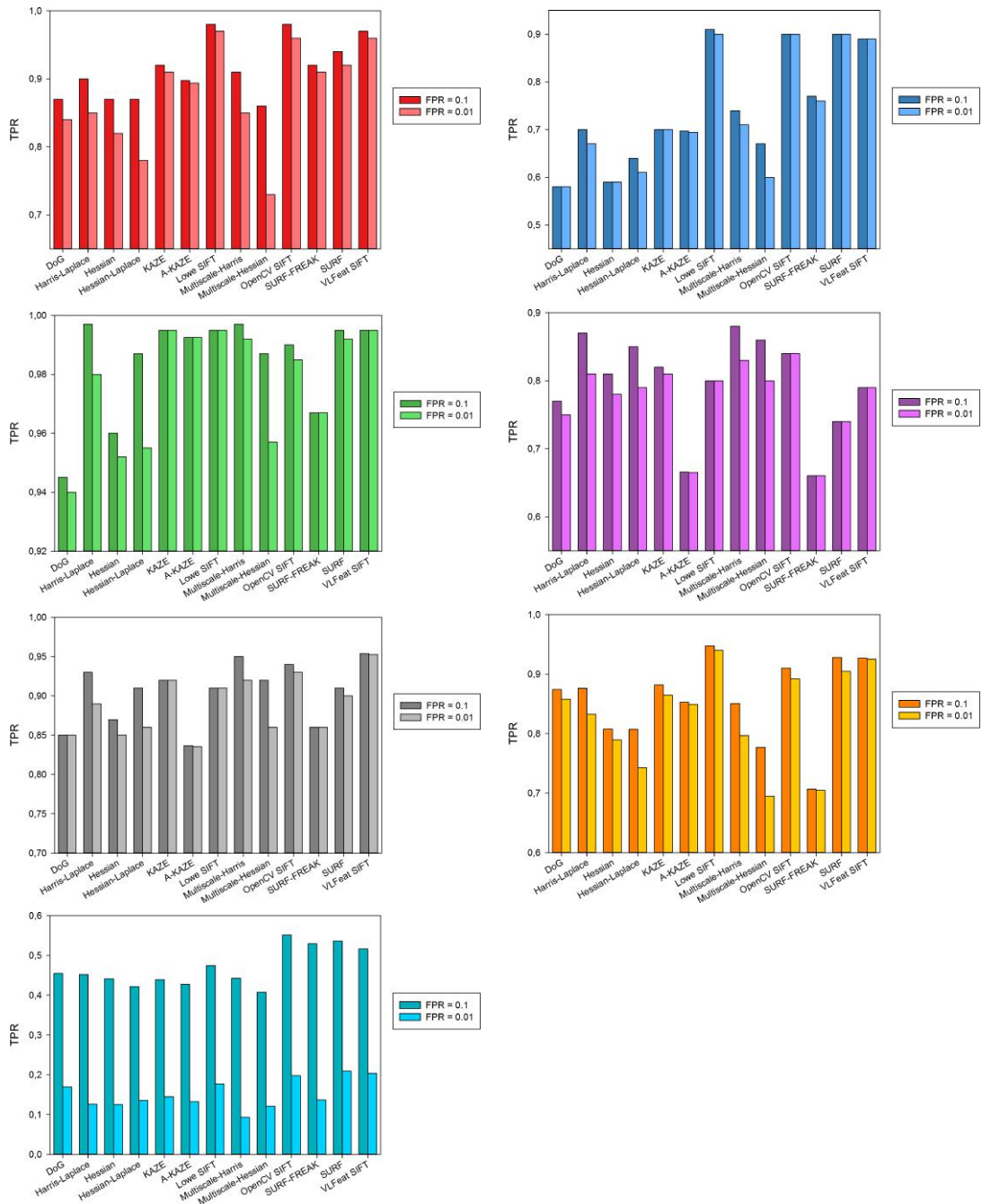


Fig. 3.20 TPR levels of all the gray-scale algorithms at a FPR of 10% and 1%. No limit on keypoints number. Top to bottom: dataset 1 (mixed text + graphics) and 2 (museum paintings), dataset 3 (video frames) and 4 (buildings and landscapes), dataset 5 (common objects and scenes) and 6a (SuperMarket Milan), dataset 6b (SuperMarket Milan: color significant objects).

Keypoint-based Approaches

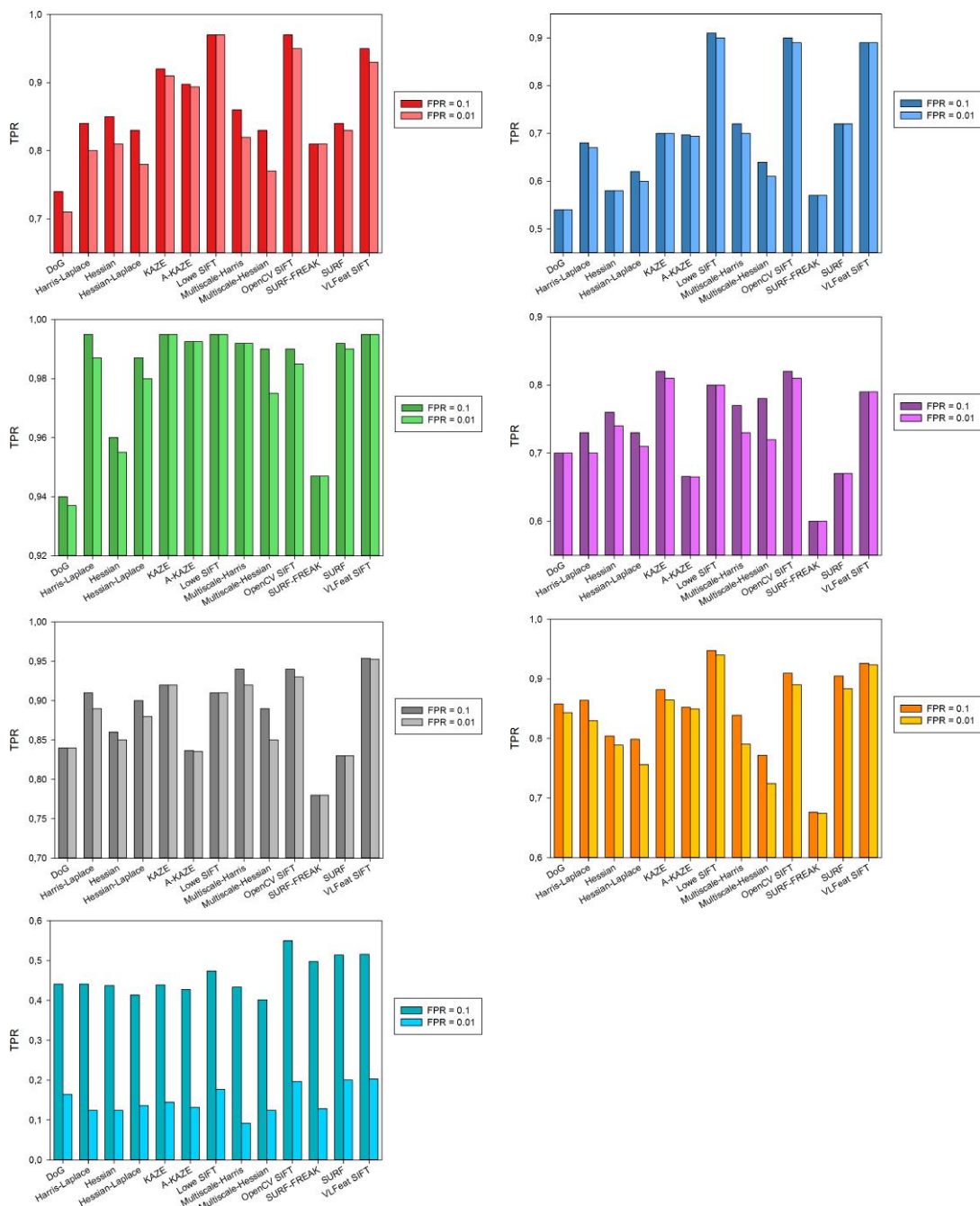


Fig. 3.21 TPR levels of all the gray-scale algorithms at a FPR of 10% and 1%. Keypoints number limited to 1024. Top to bottom: dataset 1 (mixed text + graphics) and 2 (museum paintings), dataset 3 (video frames) and 4 (buildings and landscapes), dataset 5 (common objects and scenes) and 6a (SuperMarket Milan), dataset 6b (SuperMarket Milan: color significant objects).

3.4 Quantitative Evaluation

Algorithm	Peak Threshold	Number of Points	Comput. Time (s)
DoG	3.0e2	869.6	0.981
Hessian (VLFeat)	3.0e2	1061.7	1.074
Hessian Laplace (VLFeat)	3.5e2	1138.2	0.940
Harris Laplace (VLFeat)	1.0e4	1068.3	1.797
Multiscale Hessian (VLFeat)	4.0e2	1565.1	0.908
Multiscale Harris (VLFeat)	2.0e4	1150.4	1.681
SIFT (VLFeat)	0.1e-2	1374.0	0.346
SIFT (OpenCV)	0.1e-2	1374.0	0.164
SURF (OpenCV)	3.0e2	987.5	0.169
SURF-FREAK (OpenCV)	0.1e-2	1321.0	0.483
Opponent SIFT (OpenCV)	0.1e-2	1374.0	0.480
KAZE (original v.1.3)	0.1e-3	1184.6	1.204
A-KAZE (original v.1)	1.0e-3	1271.3	0.112

Table 3.4 Filter response threshold value, average number of keypoints detected, and average overall time spent for detection and description. Timings have been taken on the images belonging to the Mixed text + graphics dataset (scaled at VGA resolution).

different color descriptors tested correspond to Early or Late fusion of this baseline with the corresponding color information (see Table 3.3). On most of datasets color descriptors added a little improvement in the discriminative power over the gray-levels descriptor or they performed even worse compared to the baseline. Results are reported in Fig. 3.22 and 3.23, where a dashed line represents the TPR of the baseline descriptor at a FPR of 1%. It is possible to notice that on dataset 3 (“Video Frames”) no descriptor achieved better results than the baseline. This is due to a lack of invariance of the color descriptors with respect to noise, blurriness, and other kinds of image transformations heavily present in this dataset. On dataset 1 (“Mixed text + graphics”) and 4 (“Buildings”) only a few color descriptors show better results with respect to the baseline. This is due to the fact that in some images from these datasets, color descriptors are misled by wrong white-balance correction, different lighting conditions and presence of shadows. Figure 3.24 shows some examples of image pairs having large difference in color appearance due to imaging conditions. Very good results were obtained by some color descriptors on datasets 5, 6a and 6b. Dataset 5 (“Common objects and scenes”) was composed of very colored objects, thus, the color information become highly discriminant. On this dataset, the algorithms achieving the best performances were those using a Late Fusion approach, which have a lower complexity than the Early fusion ones. On dataset 6b (“SuperMarket Milan” with

Keypoint-based Approaches

color significant objects), as expected, the gray-levels SIFT baseline performed very low because different objects with different color but the same shape were recognized to be the same object. All color descriptors algorithms achieved better results for the highest level of FPR. Limiting the number of keypoints to 1024 has almost no effect on all datasets with the exception of dataset 1 on which however the loss in TPR is always lower than 0.03.

Note that in all datasets the Transformed Color algorithm and the RGB algorithm achieved the same results. That is because all the invariance properties of the Transformed Color space are implicitly included in the SIFT algorithm itself as Van de Sande et al. claimed [134]. Differently from other domains where Opponent SIFT obtained the best results [134][21], on CDVS datasets this is not always true, especially for the dataset 3 (“Video Frames”). We speculate that this difference in performance can be explained by the fact that the acquired images are affected by distortions that cannot be completely modeled by the diagonal-offset model considered in [134].

3.4 Quantitative Evaluation

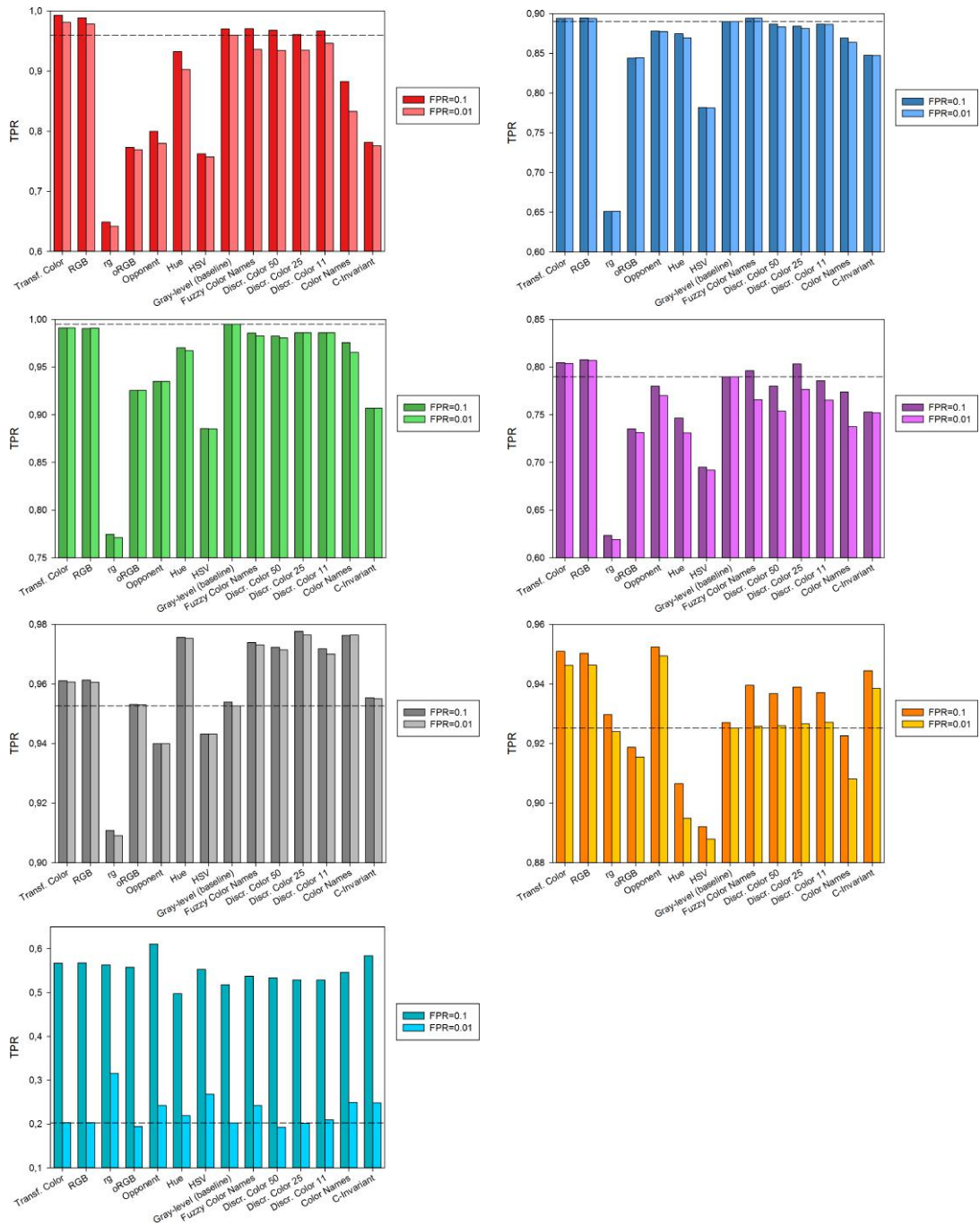


Fig. 3.22 TPR levels of all the color algorithms at a FPR of 10% and 1%. No limit on keypoints. Top to bottom: dataset 1 (mixed text + graphics) and 2 (museum paintings), dataset 3 (video frames) and 4 (buildings and landscapes), dataset 5 (common objects and scenes) and 6a (SuperMarket Milan), dataset 6b (SuperMarket Milan: color significant objects).

Keypoint-based Approaches

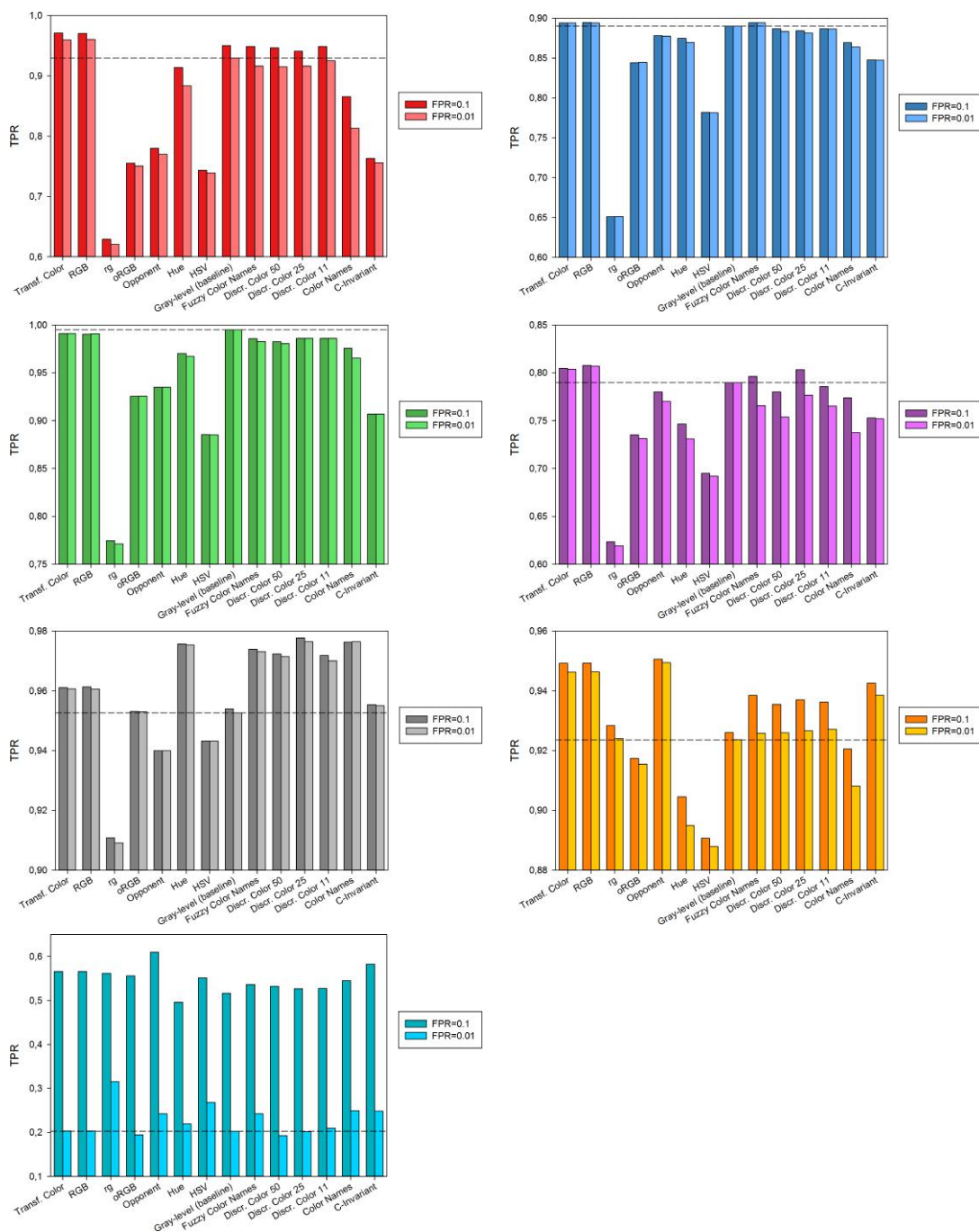


Fig. 3.23 TPR levels of all the color algorithms at a FPR of 10% and 1%. Keypoints number limited to 1024. Top to bottom: dataset 1 (mixed text + graphics) and 2 (museum paintings), dataset 3 (video frames) and 4 (buildings and landscapes), dataset 5 (common objects and scenes) and 6a (SuperMarket Milan), dataset 6b (SuperMarket Milan: color significant objects).



Fig. 3.24 Examples of images where color information can be misleading. In each image pair the reference image is shown on the left whereas the query image is shown on the right.

Chapter 4

Convolutional Neural Networks for Object Recognition

First ideas on Neural Networks are dated in the forties [89]. Since then, new models have been engineered, e.g. [77] but only a few laboratories with interests on bio-inspired neurocomputing carried on research on Neural Networks.

In 2012 the publication of a work by Krizhevsky et al. [76] showed that it was possible to train a deep neural network using GPUs to recognize thousand of categories of objects on real-world images. Their method surprisingly outperformed by a large margin all the existing approaches based on handcrafted features. Since then the interest about Neural Network increased every year and lot of laboratories started researching in that direction. Now numerous laboratories and research centers apply Deep Learning in a variety of fields different from Computer Vision like for example Computer Graphics, Automation, Stock Market Prediction, System Biology, Materials Science and several Engineering fields. Moreover an increasing number of companies are adopting Deep Neural Networks in their products or to support their production line, showing that this technology is enough mature to be adopted in many real world scenarios.

In this Chapter, first are introduced the Multilayer Perceptron and a generic Convolutional Neural Network. The second part regards algorithms to train Neural Networks including Gradient Descent and some of the most known variants. After a brief section on Data Preprocessing the list of the most used layers is presented together with some state-of-the-art network architectures. Finally two sections on Data Augmentation and Regularization close the Chapter.

4.1 Multilayer Perceptron

McCulloch and Pitts have been the first pioneers of neural networks in 1943 for introducing the idea of neural networks as computing machines [89]. However it was only in the 1958 that Rosenblatt [116] proposed the perceptron as a probabilistic model for supervised learning. It is the simplest form of a neural network and it can be used to classify linearly separable patterns. It consists of a single neuron with trainable synaptic weights and bias. Rosenblatt proved that if the input patterns are linearly separable classes, the perceptron algorithm converges and the decision surface take the position of an hyperplane between two classes. This is known as the *perception convergence theorem*.

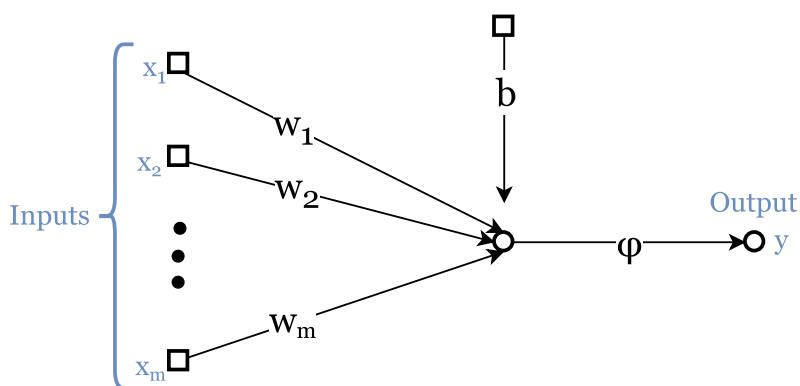


Fig. 4.1 Signal-flow graph representation of McCulloch and Pitts Perceptron model [89].

The perceptron shown in Figure 4.1 represents the model of a neuron described by McCulloch and Pitts in their work [89]. It is composed of a set of weights w_1, w_2, \dots, w_m that are multiplied by the inputs x_1, x_2, \dots, x_m plus a bias. The central node performs a sum and finally the function ϕ is an hard limiter. Examples of hard limiter functions are the logistic sigmoid $1/(1 + e^{-x})$ or the hyperbolic tangent $\tanh(x)$. Equation 4.1 is the mathematical form of the perceptron model.

$$v = \phi\left(\sum_{i=1}^m w_i x_i + b\right) \quad (4.1)$$

Weights w_1, w_2, \dots, w_m that correspond to synapses in a biological system, can be adapted (trained) on an iteration-by-iteration basis. A single perceptron is only able to discriminate linearly separable patterns.

The perceptron can, however, be used as building blocks of a larger structure called *Multilayer Perceptron* (MLP). A MLP network consists of a set of source nodes forming

the *input layer*, one or more *hidden layers* of computation nodes, and an *output layer* of nodes. The input signal propagates through the network layer-by-layer. From the learning point-of-view the task of a multilayer perceptron is to approximate a function f^* . A MLP defines a mapping $\mathbf{y} = f(\mathbf{x}; \theta)$ and learns the value of the parameters θ that result in the best function approximation.

This type of neural networks are also called *feedforward* because information flows through the function being evaluated from the input through intermediated computations used to define f , and finally to the output \mathbf{y} . As an example we can think of three functions $f^{(1)}$, $f^{(2)}$ and $f^{(3)}$ composed to obtain $f\mathbf{x} = f^{(3)}(f^{(2)}(f^{(1)}(\mathbf{x})))$ where $f^{(1)}$ is called the *first layer* of the network, $f^{(2)}$ is called the *second layer* and so on. To learn the value of parameters usually the *backpropagation algorithm* is used. It consists of two steps: a *forward pass* and a *backward pass*. In the forward pass, the predicted outputs corresponding to the given inputs are evaluated. In the backward pass, partial derivatives of the cost function with respect to the different parameters are propagated back through the network from the layer near the output to the one near the input. The network weights can then be trained using any gradient-based optimization algorithm. The whole process is iterated until the weights have converged.

4.2 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) were first presented by Kunihiko Fukushima in [43] as a tool for visual pattern recognition. In 1998 LeCun proposed a 7-level convolutional network called LeNet-5 [77] that is the most similar architecture to modern CNNs. However, only in recent years CNNs have become widespread in the scientific community, thanks to the development of high-performing architectures working on GPU [66].

A CNN is composed by one or more convolutional layers followed by nonlinear functions and optionally subsampling layers. In the last part, the CNN is usually followed by one or more fully connected layers like the Multilayer Perceptrons (see Section 4.1). The particular architecture of CNN makes them easy to train and with few parameters with respect to fully-connected networks. The reason is because CNNs exploit the 2D structure of an input image producing translation invariant features. A CNN takes an input image, which usually has undergone some minor preprocessing, processes it with a cascade of different transformation layers, to finally produce a prediction of the image class.

4.3 Training a Neural Network

Training a Neural Network consists essentially in minimizing a loss function. If the loss function and all the layers up to the input are differentiable functions it is possible to use a gradient-based method to train all the parameters in Neural Networks layers.

4.3.1 Gradient Descent

The most common optimization method used to train Neural Networks is *Gradient Descent*. It is simple to implement, efficient and powerful. The idea comes from the definition of gradient: it points to the steepest direction of the function's surface at the current point. Gradient Descent algorithm consists in choosing a random point θ_0 inside the function domain, computing the gradient of the function at that point and iteratively move on the opposite direction of the gradient until a termination condition is met. The following is the formula of gradient descent update step:

$$\theta_t = \theta_{t-1} - \alpha \nabla_{\Theta} J(\theta_{t-1}) \quad (4.2)$$

where θ_{t-1} is the current point, $J(\theta_{t-1})$ is the gradient evaluated at the current point and θ_t is the updated position in the parameters' space. α is an hyperparameter named *Learning Rate*. Carefully tuning the Learning Rate is crucial for a good convergence

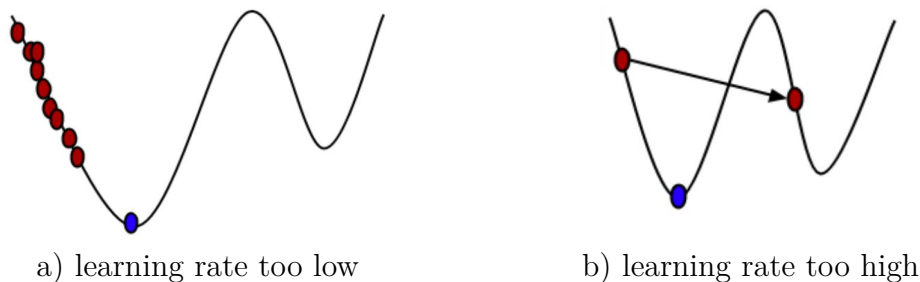


Fig. 4.2 Gradient descent steps with two different learning rates on 2D data. Figures from [UDC]

of Gradient Descent. Figure 4.2 shows the effect of a wrong tuned learning rate on a 2-dimensional solution landscape. If the value is too low it takes too many steps for the algorithm to converge to the minimum. Gradient descent finally will converge to the local optimum but it may take too many iterations and thus a long computational time. If the learning rate is too high the Gradient descent will execute large steps ending up in a different valley of the solution landscape or even climbing up the valley instead of descending it which results in divergence.

There is no guarantee that Gradient Descent converges to the global optimum. The optimum reached by the algorithm strongly depends on parameters initialization. A simple way to augment the probability of finding a good minima is to try different runs with different random initializations and keep the optimum given by the best run. Again this doesn't guarantee to reach the global optimum.

4.3.2 Stochastic Gradient Descent

From the point-of-view of a practical application the vanilla version of the Gradient Descent is used only when optimizing on toys examples. When dealing with real data usually some variants are adopted. The first is named *Stochastic Gradient Descent* usually abbreviated with *SGD*. Instead of using the whole dataset for each update, the optimization algorithm choses a random subset every step. This subset is called *Minibatch*. The assumption is that every small data subset reflect the distribution of the whole dataset thus Stochastic Gradient Descent becomes an approximation of the deterministic version with a stochasticity component given by the random order of Minibatches selection. The size of Minibatch is an hyperparameter to be chosen. The more the minibatch is small and the more the algorithm tends to pure stochasticity, the more is big and the more the approximation tends to the deterministic Gradient Descent. In real cases Minibatch size is chosen to be big enough to fit system memory. With the use of GPUs and highly parallel matricial operations the use of Minibatches can result in a huge speed-up compared to the pure stochastic Gradient Descent.

4.3.3 Momentum and other variants

During the years a lot of different variants to the vanilla Gradient Descent have been proposed. Here are reported some of the most used. The first variant is named *Gradient Descent with Momentum*. It is widely used because it is simple to implement, it cause a negligible computational overhead and it can really speed-up convergence in most cases. It is easy to illustrate the contribution of Momentum to the vanilla Gradient Descent using a metaphore. Gradient Descent can be compared to a man steadily walking down the mountain side. Gradient Descent with Momentum can be compared to a ball rolling down the mountain. The more the ball roll and the more it accelerate. That is the contribution of Momentum, adding a velocity term for parameters update.

Formally the momentum update is given by:

$$\begin{aligned}\nu_t &= \gamma\nu_{t-1} + \alpha\nabla_{\Theta}J(\Theta_{t-1}) \\ \Theta_t &= \Theta_{t-1} - \nu_t\end{aligned}\tag{4.3}$$

where ν represents the *velocity* vector and has the same dimension of the parameters vector Θ , i.e. every parameter has its own velocity. $\gamma \in (0, 1]$ is a hyperparameter allowing to control how much the current optimization step is influenced by past steps. Intuitively the use of Momentum can really speed up convergence to the minimum when the solution landscape is characterized by shallow ravines. Moreover in some cases it can be useful to exit from small ditches, i.e. local minima.

Nesterov Accelerated Gradient Following the metaphore of the ball rolling down the mountain side, a problem with Momentum arises when the ball reaches the bottom of the valley, i.e. local minimum, and it starts to roll up the opposite side of the valley for the high speed. *Nesterov Accelerated Gradient* can solve this type of problem. The underlying idea is to update the gradient step not based on the current point but on an estimation of the gradient value at the next step. Formally:

$$\begin{aligned}\nu_t &= \gamma\nu_{t-1} + \alpha\nabla_{\Theta}J(\Theta_t - \gamma\nu_{t-1}) \\ \Theta_t &= \Theta_{t-1} - \nu_t\end{aligned}\tag{4.4}$$

The main difference is on term $\nabla_{\Theta}J(\Theta_t - \gamma\nu_{t-1})$. Now the gradient is not computed anymore on the current point but on a projection of the next Gradient Descent step.

There are other well known optimizers widely used in the Deep Learning community that strenghten their update step by taking into account past history of parameters' updates. Here is a non-exhaustive list: Adagrad [39] where the idea is to base the update step on the squared roots of past steps. Adadelta [148] that tries to overcome the problem of collecting too many past steps with the consequent decay of influence on the current step. Adam [73] that keep track of past update steps and Momentum. All the names of these optimizers start with prefix "Ada", contraction of "Adaptive" because they store information about gradient steps for each parameter being in fact adaptive optimizers.

4.4 Data preprocessing

Preprocessing and normalizing input data is a crucial step in every machine learning pipeline [53], [99], [55]. Figure 4.3 shows a bidimensional feature space with data labels distributed in two classes. Yellow and green areas represent the plane classified by a decision tree. The simple classifier can put the division boundary between the two classes when data is decorrelated. In the Neural Network community three types of

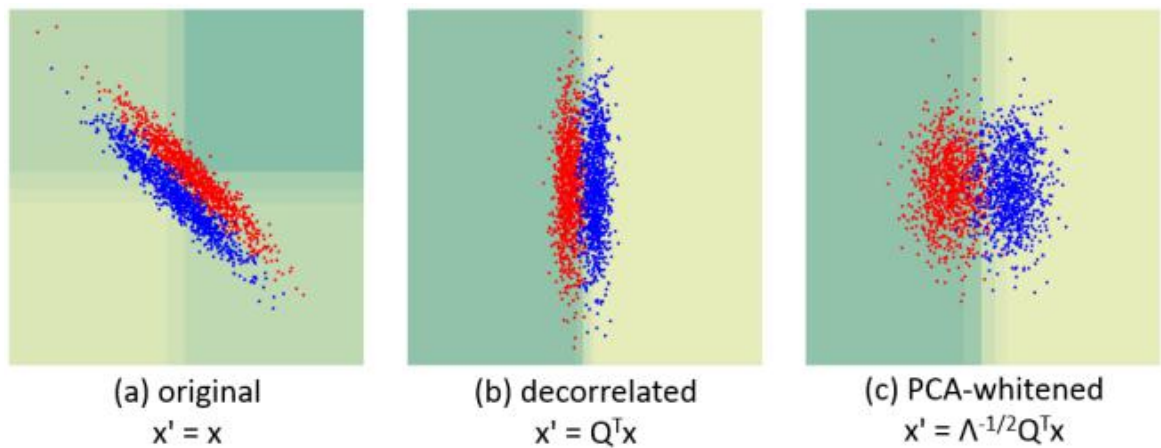


Fig. 4.3 Decorrelating features is critical. Classification trees with both decorrelated and PCA-whitened features. Figure from [99]

data preprocessing are usually adopted:

- **Mean subtraction.** It involves the computation of the mean for each input feature independently and the subtraction of the resulting value for all the input data. If the input is an RGB image, each channel is normalized independently.
- **Contrast Normalization.** This is one of the most used form of preprocessing because it is computational inexpensive but effective. It consists in subtracting the mean and dividing by the standard deviation each input feature independently, i.e. image channels. The networks for logo recognition described in Chapter 5.2 make use of this preprocessing step. In Section 5, among the experiments for the improved logo recognition pipeline, are presented some experiments on the use of Contrast Normalization as input processing step. Table 5.7 shows that Contrast Normalization improves Neural Network performances by a consistent margin on multiple experiments.
- **PCA and Whitening** PCA involves the mean subtraction first to obtain zero-centered data. The second step needs the computation of a covariance matrix.

Convolutional Neural Networks for Object Recognition

Then the eigenvectors of the covariance matrix are used to project the data space into a new space where features are less correlated. This helps the learning process especially with simple classifiers, e.g. classification trees, as an example see Figure 4.3. Another related transform is called *Whitening* and it consists in dividing every dimension by the eigenvalue to normalize the scale. While the geometric interpretation of PCA is a rotation of the feature space, whitening represent a scaling operation. This transformations are reported here for completeness but are not used as preprocessing step in the next sections. They are not very common in the Deep Learning community. The reason is that they can be very expensive to compute on big datasets and they usually do not provide a significant improvement in performances on complex models.

4.5 Common CNN Layers

In the following paragraphs the most common layer composing a CNN are presented. Specific architectures may use their own types of layers. Some layers or modules are introduced in Section together with their architectures.

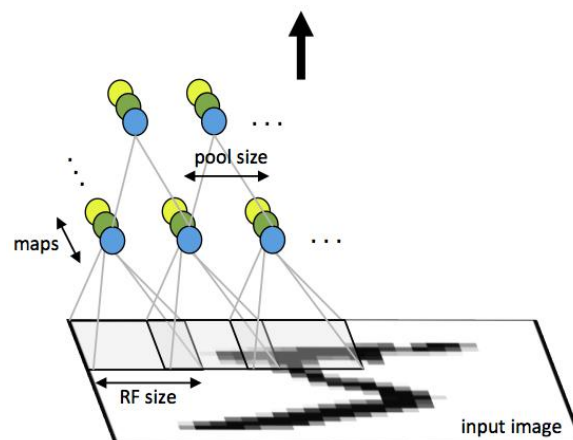


Fig. 4.4 First two layers of a Convolutional Neural Network. The nearest to the image is a Convolutional layer whereas the farthest is a pooling layer. Image from UFLDL Stanford tutorial [UFL].

Convolutional layers are the main type of layer used in CNNs. They are designed to perform a convolutional operation on the input data, which can be either the original image or the result of hidden layers. The kernel involved in this operation is not hand-encoded, but automatically learned through the backpropagation algorithm used

to train the network. The input to a convolutional layer is a $m \times m \times r$ image where m is the height and width of the image and r is the number of channels. The convolutional layer have k filters of size $n \times n \times q$ where n is smaller than the dimension of the image and q can either be the same as the number of channels r or smaller. The size of the filters gives rise to the locally connected structure which are each convolved with the image to produce k feature maps of size $m - n + 1$. Each map is then subsampled typically with mean or max pooling over $p \times p$ contiguous regions where p is usually 2. Figure 4.4 shows a scheme of the first two layers of a Convolutional Neural Network, i.e. Convolution and Pooling.

Non linearities are implemented in the CNN by activation functions. The most used activation function is ReLU (Rectified Linear Units) [76]. It keeps only the positive part of the input without applying any kind of upper bound to the signal. This prevents neurons saturation and promotes feature sparseness.

$$f(x) = \max(0, x) \quad (4.5)$$

Other non-linear activation functions commonly used are the logistic sigmoid $1/(1+e^{-x})$ or the hyperbolic tangent $\tanh(x)$. Krizhevsky et al. showed in [76] that a deep CNN with ReLU activation functions converges six times faster than a deep CNN equipped with sigmoids. Figure 4.5 shows a comparison of the three activation functions. While Sigmoid and Tanh has lower and upper bounds, ReLU puts only a lower bound to the input signal.

Pooling works as average or maximum filters, and as such are used to reduce the impact of small variations in the signal. Furthermore, they allow for a dimensionality reduction of the input data. Their effect depends on the filter size and stride. Figure 4.4 shows a scheme of the first two layers of a Convolutional Neural Network, i.e. Convolution and Pooling. The receptive field size, RF size in the picture, is given by the filter size.

Dropout [126] are used to reduce overfitting, which can occur when the number of training examples and number of CNN parameters are unbalanced, with the second one being greater than the first one. This is done, at every training iteration, by randomly dropping some activations with probability p . At testing time instead, all the neurons are used, but their responses are weighted by p itself. This ensures the output at test time to be the same as the expected output at training time.

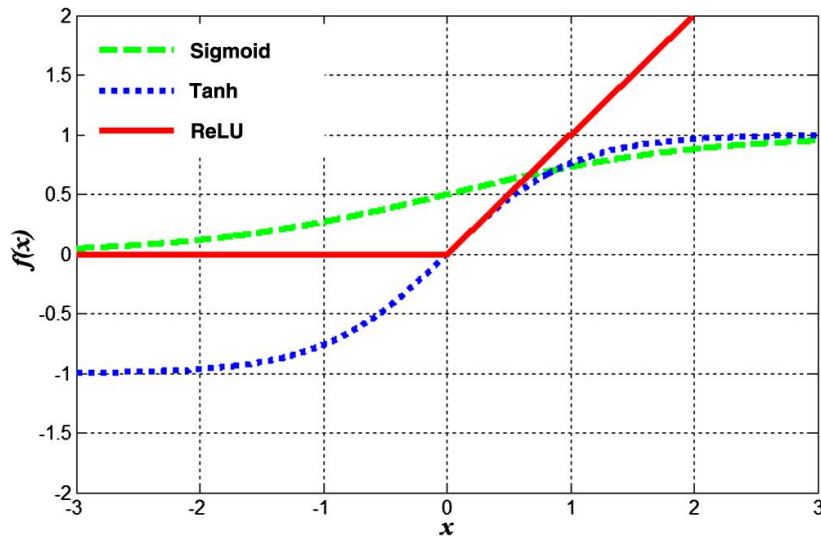


Fig. 4.5 Three most used non-linear activation functions. Sigmoid: $1/(1+e^{-x})$, Tangent: $\tanh(x)$ and Relu: $f(x) = \max(0, x)$. Figure from [144].

Batch Normalization It has been introduced in 2015 by Ioffe et al. [62] to reduce the *Internal Covariate Shift* and thus to speedup Deep Neural Networks training. In Section 4.4 it has been pointed out the importance of normalizing input data. In Deep Neural Networks, during training, the input distribution of every layer drastically changes at every forward pass due to the evolving weights of the previous layers. This effect is amplified the more the input signal goes through the network layers. This makes impractical to train very deep networks and make the training process very sensitive to the choice of hyperparameters. Batch-Normalization has been introduced to overcome this problem. The input of every layer is normalized channelwise with respect to minibatch statistics. Mean and standard deviation of the minibatch are used as estimators of the dataset statistics. The following equation formally express the Batch-Normalization layer:

$$\hat{x}_i = \frac{x_i - \mu_b}{\sqrt{\sigma_b^2 + \epsilon}} \quad (4.6)$$

$$y_i = \gamma \hat{x}_i + \beta$$

where x_i is a single activation value given as input to the Batch Normalization. μ_b represents the mean of the activations computed over the current mini-batch $\mu_b = \frac{1}{m} \sum_{i=1}^m x_i$, and σ_b^2 is the mini-batch variance. y_i represents the output of Batch Normalization. It is the result of applying an affine transformation with parameters γ and β to the normalized output. γ and β in Batch Normalization layers are learnt by

the network via backpropagation.

Batch Normalization acts implicitly as a regularizer, see Section 4.9 for more details.

Loss function It is the most fundamental layer in a Neural Network. It is usually the last layer after the network output and it measures the error between the network output and the groundth. In the next sections are faced mostly classification problems, thus here are reported two commonly used loss functions for classification.

The first introduced here is named *Cross-Entropy Loss*. Here is the formula:

$$L_i = -\log\left(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}\right) \quad (4.7)$$

where L_i is the loss function value for each sample and the total loss value is given by $L = \frac{1}{N} \sum_i L_i$ where N is the number of samples. f_{y_i} is the activation of the last output layer of the Neural Network before the loss function. The denominator normalizes the output by summing over all the activations.

Another common used loss function for classification problems is named *Hinge Loss* and it is formulated as follows:

$$L_i = \sum_{j \neq y_i} \max(0, f_j - f_{y_i} + 1) \quad (4.8)$$

Hinge Loss it is alternatively named SVM loss because it is mainly used to train Support Vector Machines. The idea is to set a margin and penalize predictions very far from the decision boundary. It doesn't promote a high difference between the correct activation and all the others like Cross-entropy loss does.

4.6 State-of-the-art CNN architectures

Many CNN architectures that make use of the above building blocks have been proposed in recent years. Although many variants have been investigated, in this Section, only four most known architectures are presented.

All the models presented here won the Imagenet Largescale Visual Recognition Challenge (ILSVRC) competition [121] trough the years. ILSVRC was the first object recognition and localization competition over a large set of images, i.e. 15 millions involving more than 22k classes. For its huge cardinality and high-quality finegrained annotations, ILSVRC has become the standard benchmark for object recognition Deep CNN architectures. Architectures are presented in chronological order:

4.6.1 AlexNet (2012)

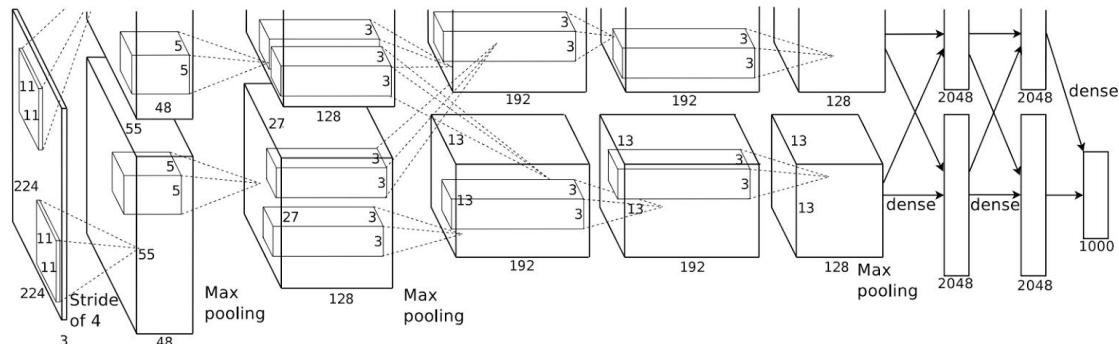


Fig. 4.6 AlexNet CNN architecture. Figure from [76]. The delineation of responsibilities between the two GPUs is shown. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. Numbers under blocks represents channels for each layer. Input is left and output is right. Note the increasing numbers the more the layers are far from the input.

Proposed by Krizhevsky et al. [76]. It was the first Deep CNN to win the ILSVRC competition [121] in 2012. It can really be considered one of the most revolutionary works in both fields of computer vision and machine learning. For the first time, it had been shown that Neural Networks could be efficiently trained on huge datasets (15M images) with higher accuracy than handcrafted methods. It was one of the first Deep CNNs implemented on GPUs together with the network presented by Cireřan et al. [33]. The structure of the two networks is similar. The main difference is that [33] has been tested on relatively small datasets with low-resolution images. Krizhevsky et al. to train on higher resolution images proposed a training system using two GPUs that communicate only between certain layers to maximally reduce communication overhead. In Figure 4.6, for each layer, are shown two big blocks that represent convolution layers distributed on two GPUs. Dotted lines between blocks represent communication between layers. Some of them are cross-GPUs.

The network is composed by 5 convolutional layers followed by ReLUs and max pooling layers. The final part of the network is composed by 3 fully connected layers. The use of ReLU nonlinearities between both Convolutional and Fully-connected layers was another important ingredient for fast convergence.

The whole network has 60 millions parameters and 650k neurons, a much higher number with respect to the training dataset cardinality, i.e. 15M. To deal with overfitting, different techniques described in Section 4.9 were adopted. They performed data

augmentation by rotating and flipping images and applying lighting noise. They also made use of Dropout layers and Local Response Normalization. Figure 4.6 shows the Alexnet structure made of stacked Convolutional, Pooling layers and Fully-connected (dense) layers.

4.6.2 VGG (2014)

This architecture has been proposed by Simonyan and Zisserman [124] and scored first at ILSVR-2014 challenge for top-5 error. The main idea is to substitute large receptive-field convolution layers like 5x5, 7x7, 11x11 with small kernels. The same receptive field of a 5x5 kernel filter can be obtained by stacking two 3x3 kernel layers with a nonlinear function inbetween. This simple trick allows to reduce the number of parameters while at the same time maintaining the receptive field size. Furthermore the introduction of more nonlinearities empowers model capacity. Authors didn't employ Local Response Normalization because from experimental results it brought no improvement while slowing down training and increasing model size. They proposed five different VGG architectures with increasing depth. The shallower architecture presents 8 convolutional layers w.r.t 8 of the AlexNet while the deeper one have 16 convolutional layers. Network parameters vary from 133M for the 8 Conv version to 144M for the 16 Conv version. Note that the increase in parameters by adding convolutional layers is negligible. The difference is made by the fully-connected layers that have 4096 neurons each, instead of 2048 of AlexNet architecture.

VGG networks are still used as feature extractors in many state-of-the-art architectures. In Appendix A is presented a way to perform texture synthesis and artistic style transfer using Deep Neural Networks. The system makes use of pretrained siblings VGG networks. The style transfer architecture has been first investigated by Gatys et al. in [47]. The use of siblings VGG networks has been then exploited by Johnson et al. in [67] as a new type of loss function called *Perceptual Loss*. Perceptual Loss is now widely used in many state-of-the-art generative models.

4.6.3 Inception (2014)

The first Inception network known as Inception v1 has been introduced in 2014 by Szegedy et al. [129]. This version won the ILSVRC 2014 submission with the name GoogLeNet. The name is an homage to the first convolutional network by LeCun et al. [77]. The network is mainly composed by stacking *Inception modules*. Each inception module is composed by different convolution kernels of various sizes. The

input is processed by each kernel and the output is concatenated. Figure 4.7 shows a naïve version of the Inception module. Another version of the Inception module perform dimension reductions before the 3x3 and 5x5 convolutions with 1x1 filters. Another novelty introduced in Inception v1 was the use of auxiliary classifiers to

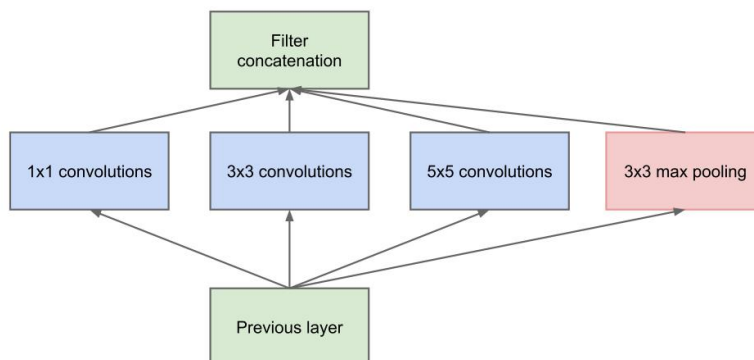


Fig. 4.7 Inception module naïve version. Input is processed through filters with different kernels and the output is concatenated. Figure from [129].

help convergence during training. A small network branched after each Inception module and perform classification on 1000 classes. These auxiliary classifiers were then discarded at test time. From 2014 different versions have been released almost every year. v2 use stacked smaller 3x3 convs instead of 5x5 and 7x7 kernels. v4 and v5 includes novelties from other architectures like Residual Blocks presented in the next paragraph.

4.6.4 Resnet (2015)

Deep Residual Network (Resnet) has been published in 2015 by He et al. [58] and won numerous ILSVRC-2015 challenges on detection and localization tasks. The novelty of the architecture relies on a *Deep Residual Learning* framework. Each residual block is trained to fit a residual mapping function and multiple residual blocks are stacked. A residual block can be realized by a feedforward neural network with *Shortcut connections*, i.e. they are simple functions that can skip one or more layers. In the case of Residual Networks the shortcut connections simply perform identity mapping and their outputs are added to the outputs of the stacked layers. The structure of a Residual Block is shown in Figure 6.2. It is composed by two convolution layers with a non-linear activation function, i.e. ReLU in between. Every convolution is followed by a Batch Normalization layer [62]. The input signal flows through the Residual Block

and at the same time pass through the skip connection. It is then summed up with the output from the Residual Block itself. The network is composed by stacking several

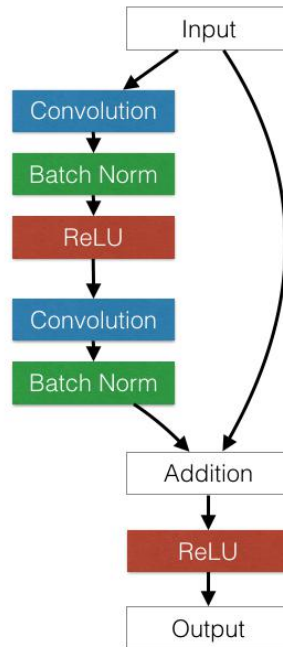


Fig. 4.8 Residual Block as implemented in Residual Networks [58]. Figure from [res]

Residual Blocks. Some blocks perform a pooling operation in order to reduce the spatial support of features. Only the last fully-connected that maps directly to the output classes is retained. All the other fully-connected layers are discarded in order to reduce the number of parameters. The Residual Learning framework Networks is capable of handling very deep architectures. Resnet authors reported experiments with different depths discovering that it is possible to successfully train a 1202-layers network. However a comparable test error can be obtained with a 152 layers network and that is the type of network used to win the ILSVRC-2105 competition.

Concurrent with Resnets, Srivastava et al. designed Highway networks [127], [128]. They are based on a similar principle but the main difference relies on the type of skip connections adopted. While in Resnets identity shortcuts are always open and all information passes always through, highway networks make use of gating functions to control the information flow. Since Residual Networks skip connections are parameter free they are easier and faster to train.

4.7 Network Parameters

Since AlexNet model in 2012 all the consequent models have dramatically increased their accuracy. VGG networks have raised state-of-the-art performances on Imagenet dataset [121] by 15% but consistently increasing the number of parameters and thus the computational complexity. Inception and Resnets have further pushed the boundary even surpassing human level performances [57] for the task of object recognition on Imagenet dataset. These recent architectures not only increased accuracy on different computer vision tasks but also drastically reduced the number of parameters and training time. Figure 4.9 shows a chart representing the top-1 accuracy on Imagenet vs the number of Giga-operations and the number of parameters. AlexNet from [76] shows an accuracy of almost 55% with 65M of parameters. Both VGG-16 and VGG-19 have more than 125M of parameters. Resnets and Inception networks come in different sizes ranging from 70% to 81% in accuracy and from 10M to 95M of parameters.

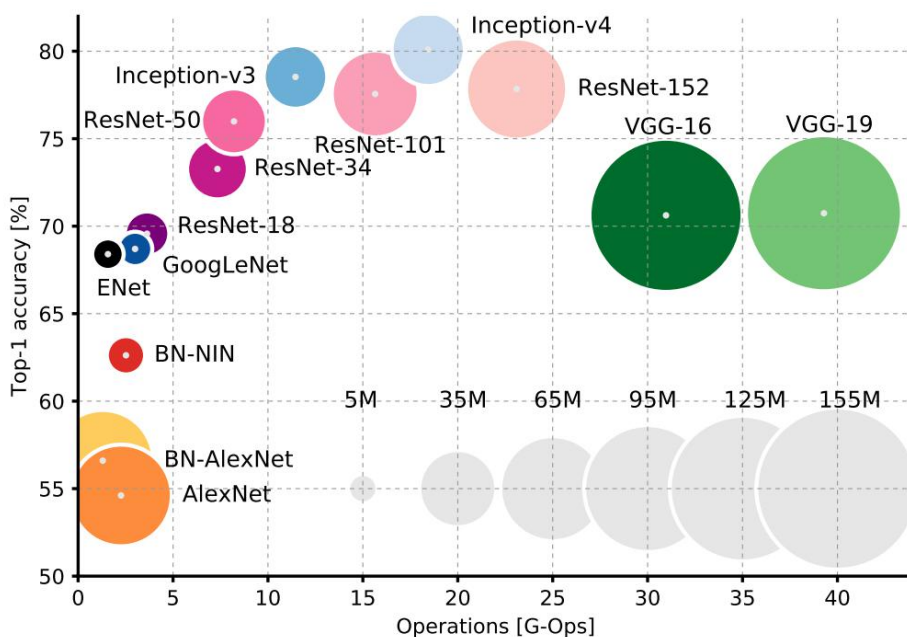


Fig. 4.9 Comparison of Deep CNN architectures. Top-1 accuracy on Imagenet for the object recognition task vs Giga-operations. The chart shows also the number of parameters for each model. Figure from [29]

4.8 Data Augmentation

The use of Deep Networks in recent years has been made possible mainly by the use of GPUs for general computations and the availability of high quantities of data. Furthermore Deep Neural Networks models trend seems to go toward a more efficient use of parameters and thus a reduction in models' size. Despite that, the high capacity of common Neural Networks make the training process prone to overfitting. Two effective methods to overcome the overfitting problem while maintaining the network capacity are Regularization and Data Augmentation. In this Section are explained some techniques used to artificially augment image datasets.

The purpose of synthetic data augmentation is to provide the model with more fake data to induce the model to a better generalization. The idea is to provide new examples with a subset of the plausible variations that can be found in the test set. However the type and degree of image transformation should be carefully designed in order to avoid to hurt the learning process. Here is reported a short list of the most common transformations:

- **Affine transforms.** This type of data augmentation modifies the spatial arrangement of pixels. Common transformations include translation, rotation, shear and scale. Section 5.6 and Figure 5.7 show this transformation type applied to logo images.
- **Flipping.** This is a very common type of geometric transform used for generic objects but not for example in paintings and logos datasets.
- **Color jitter.** It consists in a modification of contrast, brightness and saturation of the input image independently. Usually the image is first converted to an apposite color space like HSV and converted back to the RGB channels after the transformation.
- **Lighting noise.** It is a pixelwise transform based on the eigenvalues of the RGB pixel distribution of the dataset. It has been introduced by Krizhevsky et al. [76]. This transformations include *light intensity change and shift* and also *light color change and shift* as presented in Section 2.1
- **Random Noise or Gaussian Blur.** It is used to force the model to be able to handle low quality images. Figure 2.6 in Section 2 shows the visual effect of Blur and Noise on images.

From an implementation point-of-view the best option is to apply these transformations on the fly during training instead of generating an augmented dataset and then train on it. The contribution to the training process is equivalent but the latter approach produces a huge dataset to be stored. Instead, on the fly transformations, come at zero computational cost if made on CPU while the GPU is processing the Neural Network. Common Deep Learning frameworks allow this parallelism with few lines of code.

Data augmentation at test time: Query Expansion All the transformations described in the above section can also be applied at test time as a form of *Query expansion* with the exception of random noise and gaussian blur for obvious reasons. When working with object detection and classification, usually we have to deal with two kinds of variability. The *intrinsic* variability corresponds to the fact that two instances of the same object class can be visually different, even when viewed under similar conditions (e.g. different versions of a logo may differ for some details or colors). The *extrinsic* variability refers to those differences in appearance that are not specific to the object class (e.g. different viewpoints, lighting conditions, image compression artifacts). Query expansion is a simple way to handle at test time the extrinsic variability of datasets especially when the training set cardinality is low.

Section 5.4 introduces the use of Query expansion to handle the extrinsic variability of logo dataset. In Table 5.5 are reported comparisons on the use of Query Expansion.

4.9 Regularization

William of Ockham (circa 1287 - 1347) was an English Franciscan friar. He was a theologian and a scholastic philosopher. He is known for the following quote:

“Numquam ponenda est pluralitas sine necessitate.”

that can be translated as “Plurality must never be posited without necessity”. The principle expressed by this sentence is known as Ockham’s razor. It is a problem-solving principle stating that, among competing hypotheses, the one with the fewest assumptions should be selected. Regularization is an application of the Ockham’s razor in the machine learning field. The role of regularization is to discourage complex explanations of the world in favour of simpler ideas, even if they fit observations better. The reason is because usually complex explanations are unlikely to generalize well to unseen data.

Neural Networks are powerful and expressive models. Early theoretical studies proved that even a very simple feedforward architecture with only a single hidden layer

but a sufficiently large number of neurons can approximate every function [36], [42]. Neural Networks are *universal approximators*.

A recent work by Zhang et al. [149] showed that common Deep CNN models like those reported in Section 4.6 can easily memorize datasets with thousand of images without generalizing at all. Zhang et al. in their experiments substituted real labels of the CIFAR-10 dataset (see Section 3.3.3) with completely random labels discovering that modern architectures can achieve zero training error. This happens without additional time needed for convergence suggesting that the training process is still easy. Obviously, while the train error is zero, the test error is still equal to random chance confirming the fact that the model is not able to generalize. They tried different types of randomization experiments: increasing incrementally the quantity of random labels, random shuffling image pixels, substituting input images with gaussian noise. All these experiments suggest one thing: common Deep CNN models have huge effective capacity. They are large enough to shatter the training data and rich enough to memorize the whole training set. They also mathematically prove the existence of a theorem:

Theorem 1. *There exists a two-layer neural network with ReLU activations and $2n + d$ weights that can represent any function on a sample of size n in d dimensions.*

the proof is given in the Appendix of [149]. This represents a much stronger claim about Neural Networks capacity showing that, not only huge models with millions of parameters can completely overfit, but even small Neural Networks with two layers and a precise number of neurons can memorize any dataset.

Implicit and explicit regularization is the way to avoid overfitting and promoting generalization. Here are listed the main forms of implicit and explicit regularization techniques used in Deep CNNs:

- **Network structure.** The choice of the network architecture is the most incisive form of implicit regularization. The use of shared weights or local connections drastically reduce the number of parameters of the network. Convolutional layers exposed in Section 4.5 are very effective on data with stationary properties like images, 1d signals etc. Convolutional layers can be seen as layers of neurons with both properties: shared weights and local connections. Intuitively deeper architectures tend to act as regularizers over wide ones. Higher classification accuracy of VGG network structure with respect to AlexNet, in Section 4.6, mostly is due to the use of a deeper architecture in which 5x5 filters are substituted with two layers of 3x3 filters interleaved by a non-linearity. The receptive field of

the two layers is still 5x5 but the benefit is double: less parameters but higher capacity.

- **Early stopping.** This is another implicit form of regularization. It consists in monitoring train and test errors throughout the learning process and stopping it when the test error stabilize or increases. It is widely used because it is cost free to implement and use and it can be used jointly with all the other regularization methods.
- **l_2 -regularization and weight decay** It is an explicit form of regularization. It is widely known and used in the machine learning community. *l_2 -regularization* consists in adding a term in the current loss function to penalize high values of weights. *weight decay* consists in setting to zero low values of weights. It is introduced together here because it can be used together as a way to force weight sparseness. In general every way to induce sparseness is a form of regularization.
- **Dropout** It has been introduced by Srivastava et al. [126] as a regularization technique to prevent coadaptation of neurons. It is simple to implement and use. During training, in the forward pass, weights are set to zero with a certain fixed probability where as, at test time, dropout is disabled. These neurons are *dead* for a single forward/backward pass forcing the strength of neighbor neurons. The noise introduced in the following layers help the network to be more robust to differences in the input. Authors of [126] state that a network trained with Dropout can be seen as an ensemble of multiple instantiations of the same model. The output at test time is thus a consensus of the ensemble. A well-known variant of Dropout made for the same purpose is Dropconnect [143].
- **Batch Normalization** Introduced in 2015 by Ioffe et al. [62] to reduce the *Internal Covariate Shift* and to speedup Deep Neural Networks training. See Section 4.5 for a more detailed explanation. Normalization statistics applied to every example highly depend on the current minibatch thus input to every layer it is no more deterministic. This stochasticity acts as regularization allowing to reduce or removing other explicit regularizers like Dropout or l_2 -regularization. Very deep models like Resnets, presented in Section 4.6, wouldn't have been possible without Batch Normalization layers and they do not make use of any other form of explicit regularization during training.
- **Multitask learning** It is the practice of training a single model to make multiple types of predictions. It is known that multitask learning introduce an inductive

bias and reduce the Rademacher complexity of the model, i.e. its ability to fit random noise [120]. For extensive experiments on the ability of Neural Networks to fit random noise see [149]. Deep Network described in Section 6.2.3 is a multitask neural network trained to predict from the same painting with a single forward pass three attributes: author, style and genre. It is trained to perform three different tasks. Table 6.5 empirically shows that the same model trained to perform multitask predictions is less prone to overfitting and thus obtain higher accuracy. This holds, intuitively, when tasks are somehow correlated. When performing multitask training on uncorrelated tasks, this can even hurt performances as shown in Table 6.5 with the task of Genre recognition. While style and author of paintings are highly correlated, genre is not.

Chapter 5

Logo Recognition

Logo recognition in images and videos is the key problem in a wide range of applications, such as copyright infringement detection, contextual advertise placement, vehicle logo for intelligent traffic-control systems [111], automated computation of brand-related statistics on social media [44], augmented reality [54], etc.

Traditionally, logo recognition has been addressed with keypoint-based detectors and descriptors [13, 74, 68, 90]. For example Romberg and Lienhart [114] presented a scalable logo recognition technique based on feature bundling, where individual local features are aggregated with features from their spatial neighborhood into Bag of Words (BoW). Romberg et al. [115] exploited a method for encoding and indexing the relative spatial layout of local features detected in the logo images. Based on the analysis of the local features and the composition of basic spatial structures, such as edges and triangles, they derived a quantized representation of the regions in the logos. Revaud et al. [113] introduced a technique to down-weight the score of those noisy logo detections by learning a dedicated burstiness model for the input logo. Boia et al. [26, 25] proposed a smart method to perform both logo localization and recognition using homographic class graphs. They also exploited inverted secondary models to handle inverted colors instances. Recently some works investigating the use of deep learning for logo recognition appeared [40, 61]. Eggert et al. [40] investigated the use of pretrained Convolutional Neural Networks (CNN) and synthetically generated data for logo recognition, trying different techniques to deal with the limited amount of training data. Also Iandola et al. [61] investigated a similar approach, proposing and evaluating several network architectures. Oliveira et al. [102] exploited pretrained CNN models and used them as part of a Fast Region-Based Convolutional Networks recognition pipeline. Given the limited amount of training data available for the logo recognition task, all these methods work on networks pretrained on different tasks.

5.1 Datasets

5.1.1 FlickrLogos-32 Dataset

FlickrLogos-32 dataset [115] is a publicly-available collection of photos showing 32 different logo brands. It is meant for the evaluation of logo retrieval and multi-class logo detection/recognition systems on real-world images. All logos have an approximately planar or cylindrical surface. For each class, the dataset offers 10 training images, 30 validation images, and 30 test images. An example image for each of the 32 classes of the FlickrLogos-32 dataset is reported in Figure 5.1.

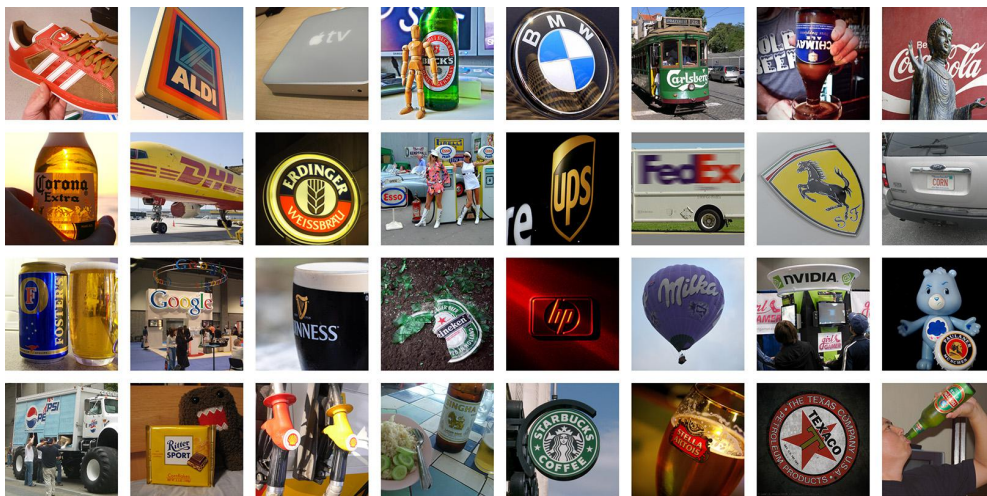


Fig. 5.1 Example images for each of the 32 classes of the FlickrLogos-32 dataset.

5.1.2 Logos-32plus Dataset

Logos-32plus [20] dataset is an expansion of the trainset of FlickrLogos-32. It has the same classes of objects as its counterpart but a larger cardinality (12312 instances). It has been collected for three main reasons: first, it is a suitable dataset size to train and test deep learning approaches. Second, Logos-32 dataset is not very representative of a data distribution for most real-world problems. Third, synthetic data augmentation is not enough to model actual logo appearance variability. The Logos-32 dataset was collected with the aim to train keypoint-based approaches. Therefore the selection of images followed some implicit guidelines, such as: most of the images are on focus, no blurry or noisy images, and usually images with highly saturated colors. As a result, the variability of this dataset mainly resides on the amount of intraclass affine transformations which can be handled very well by keypoint-based detection methods.

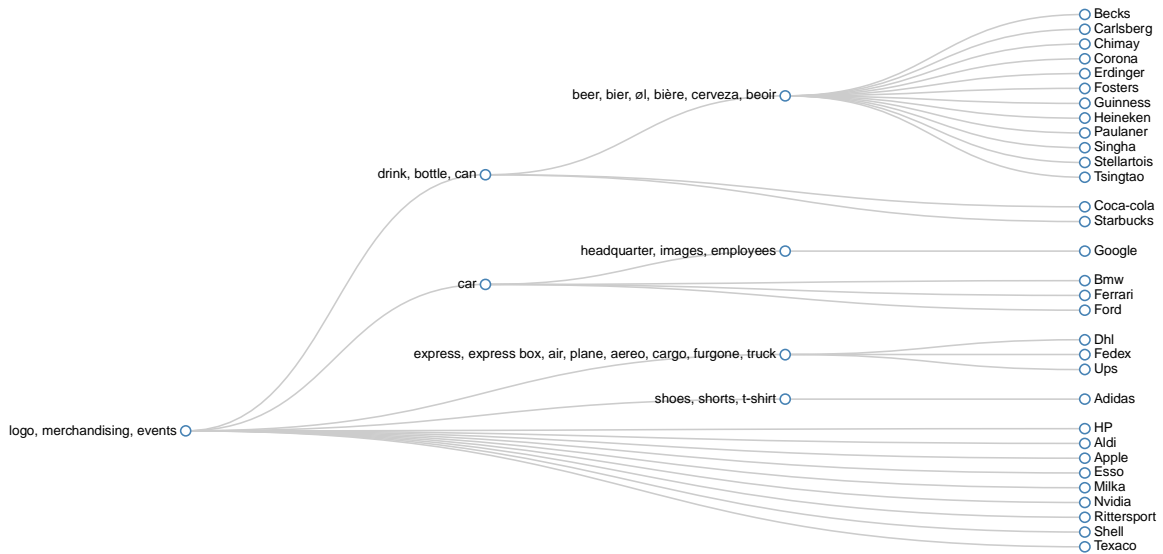


Fig. 5.2 Dendrogram representing the queries composition used to download the Logos-32plus dataset. To retrieve images of becks logos we used for instance: “logo Becks”, “merchandising Becks”, “drink Becks”, “bottle Becks”, “beer Becks” etc.

This new dataset has been collected with the aim of taking into account a larger set of real imaging conditions and transformations that may occur in uncontrolled acquisitions.

Logos-32plus dataset has been built with images retrieved from both Flickr and Google image search. In particular, to increase the variability of data distribution, multiple queries have been performed for each logo. The dendrogram scheme in Figure 5.2 shows the tags used to compose the search queries used. To compose a single query, one leaf (a single logo) has been concatenated with a single tag of an ancestor node. The whole set of queries for each logo can be obtained by concatenating the logo name (leaf) with each tag contained in all the ancestors nodes. For example, all the queries used to search for the “Becks” logo are: “logo Becks”, “merchandising Becks”, “events Becks”, “drink Becks”, “bottle Becks”, “can Becks”, “beer Becks”, “bier Becks” etc.

The dataset contains on average 400 examples per class, with each image including one or multiple instances of the same class. The detailed distribution of classes is shown in Figure 5.4 and a comparison between the FlickrLogos-32 and the Logos-32plus datasets is presented in Table 5.1. The dataset is made available for research purposes at <http://www.ivl.disco.unimib.it/activities/logo-recognition>.

5.1.3 Duplicates Removal

To ensure a high variability of the new dataset and to avoid any overlap with the existing one, we performed a semi-automatic check for duplicate images within the Logos-32plus dataset itself and with the FlickrLogos-32 dataset. The process has been carried out in two steps. First, we automatically found and discarded image duplicates using the SSIM measure [145]: we checked for similarity every pair of images within the Logos-32plus dataset itself and with the FlickrLogos-32 dataset using the SSIM measure. Images with SSIM measure over 0.9 have been discarded.

As a second step, we removed near duplicates in a semi-automatic manner. We say that two images are near duplicates if they depict the same scene with small differences in appearance with a particular focus on the portion of the image containing the logo. Examples of near duplicates are different overlapping crops of the same photo or images of the same scene from a different point of view. An interesting example of near duplicates is shown in Figure 5.3. The two images depict the same gas station from a very similar point of view. The girls in the photo are in different poses but the appearance of the Esso logo in the two images is basically the same. In detail, to remove near duplicates we used the following procedure:

- we trained our CNN (structure in Table 5.6) from scratch on Logos-32plus dataset. To accomplish this task we fed the network with crops extracted from GT annotations and Object-proposals regions.
- We truncated the learned network leaving out the last two layers (softmax and last fully-connected). This network surgery operation let us use our network as a feature extractor exploiting the robust features learned by a deep neural network. We used this truncated network to extract features from every image crop that contains a tagged logo.
- We trained a k-NN classifier on top of the extracted features (using Logos-32plus as training set) and used it to retrieve from Logos-32plus and FlickrLogos-32 the nearest five results.
- Finally we manually checked for near duplicates among the five nearest results retrieved by the classifier. All the near duplicates have been discarded from the final dataset.



Fig. 5.3 Example of near duplicates. The two images depict the same scene from a similar point of view. The appearance of the Esso logo in the two images is basically the same. We removed one of the two images from our Logos-32 plus dataset because the other one is included in the FlickrLogos-32 test set.

Table 5.1 Comparison between FlickrLogos-32 and Logos-32plus datasets

	FlickrLogos-32	Logos-32plus
Total images	8240	7830
Images containing logo instances	2240	7830
Train + Validation annotations	1803	12302
Average annotations for class (Train + Validation)	40	400
Total annotations	3405	12302

5.2 Processing Pipeline

Here is outlined a two stage processing pipeline to perform logo recognition. The idea has been first investigated by Girshick et al. [50] for generic objects recognition. Given an input image, regions that are more likely to contain an object are extracted. These regions are called object proposals. The algorithm used for the extraction of the objects proposals is class-agnostic, therefore it extracts regions of different aspect-ratios that can be used to recognize objects under different kinds of geometric transformation. These proposal are then warped to a common size (see Section 5.5) and processed for query expansion in order to increase recall. Finally a pre-trained CNN is used as

Logo Recognition

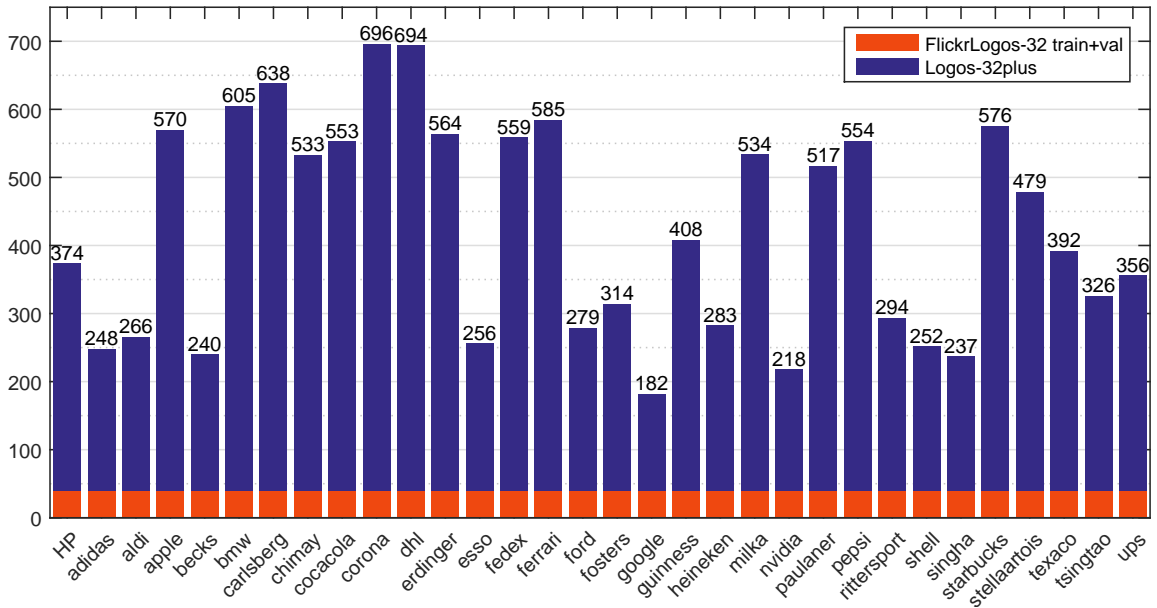


Fig. 5.4 Graphical comparison of the distribution of the 32 logo classes between FlickrLogos-32 and our augmented Logos-32plus dataset

feature extractor and a linear SVM for logo recognition and classification. Figure 5.5 shows the main steps of the recognition pipeline.

5.3 Selective Search

Selective Search has been investigated as object proposal algorithm in the logo recognition pipeline. It has been originally introduced by van de Sande et al. [136, 132]. The authors exploit a hierarchical grouping algorithm, in order to naturally generate locations at all scales, by continuing the grouping process until the whole image be-

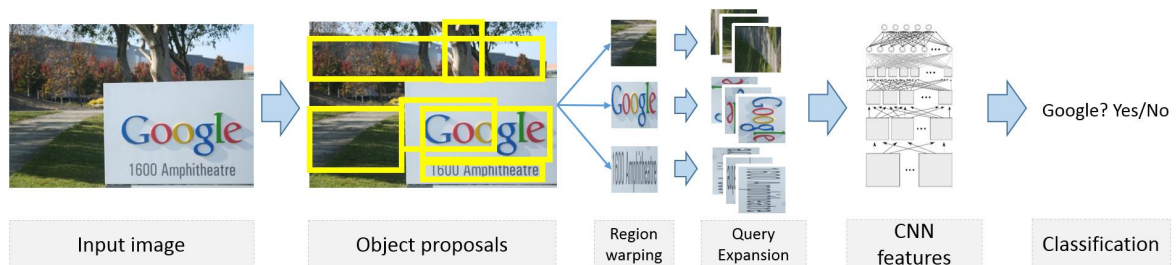


Fig. 5.5 Outline of the recognition pipeline: (1) Candidate objects regions are extracted from the image. (2) Regions are warped to a common size and multiplied through Query Expansion. (3) CNN features are computed over each region. (4) Classification is performed using linear SVMs.

comes a single region. First, a graph-based image segmentation algorithm [41] is used to create initial regions. Then, instead of using a single clustering technique, a variety of complementary grouping criteria is used to account for as many image conditions as possible. Such criteria include color similarity, texture similarity, and measures that encourage merging of small regions and overlapping regions. The final set of candidate locations is then obtained by combining the locations of these complementary partitionings.

5.3.1 Evaluation on Flickr-32 dataset

The Selective Search algorithm [132] can extract the candidate object regions upon different color spaces. In this section is reported an evaluation of Selective Search object proposals quality using different color spaces on the FlickrLogos-32 dataset.

Hosang et al. [60] introduced a class agnostic metric to evaluate the effectiveness of an object proposal algorithm: the Recall versus IoU (Intersection over Union). It is computed by varying the IoU rejection threshold, then for each threshold value, the number of overlapping bounding-boxes is counted.

In Figure 5.6 the curves for five different color spaces are reported: HSV, Lab, rg plus the Intensity channel, the Hue channel and the Intensity channel only.

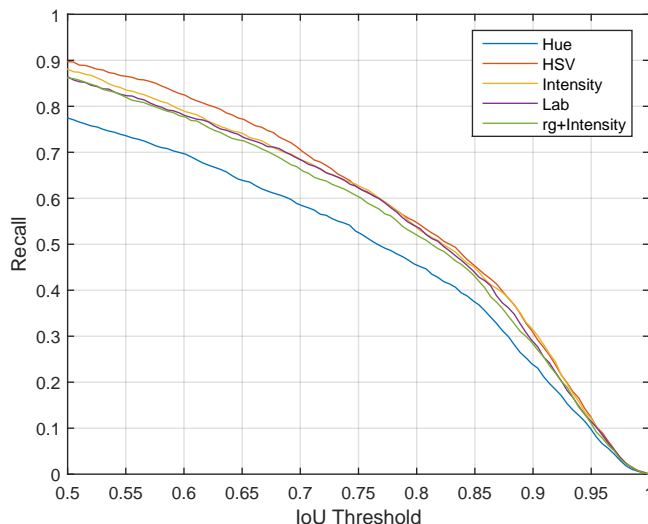


Fig. 5.6 Recall versus IoU threshold for different color spaces on Flickr-Logos dataset.

In Table 5.2 is reported the list of the mean number of object proposals extracted and the Average Recall for each colorspace tested. The Average Recall was computed for levels of IoU from 0.5 to 1.

Logo Recognition

Table 5.2 Mean Number of Object Proposals per image and Average Recall value for each color space

Color space	#Proposals/image	Average Recall
Hue	486	0.472
HSV	642	0.566
Intensity	412	0.552
Lab	292	0.545
rg + Intensity	352	0.535

Experiments in the next sections are performed with Selective Search based on the HSV colorspace because it shows the higher recall among other colorspace.

5.4 Query expansion

The FlickrLogos-32 dataset exhibits high levels of extrinsic variability. To cope with this, at test time, each candidate location extracted with Selective Search, is transformed to produce an expanded query. The candidate location is then assigned to the class with maximum confidence over all the expanded query.

5.5 CNN features

Instead of learning an ad-hoc CNN for the logo recognition problem, it is investigated how a pre-trained one works on this problem. It is in fact known that the features produced by CNNs in the last layers before the class assignment work effectively on other problems as well [123]. To this end, a Caffe [66] implementation of the CNN described by Krizhevsky et al. [76] is employed to extract a 4096-dimensional feature vector from each 227×227 RGB image. This is done by subtracting a previously computed mean RGB image, and forward-propagating the result through five convolutional layers and two fully connected layers. More details about the network architecture can be found in [76, 66]. The CNN was originally trained on a large dataset (ILSVRC 2012) with image-level annotations to classify images into 1000 different classes. Features are obtained by extracting activation values of the last hidden layer. The extracted features for each candidate location are then used as input to a Support Vector Machine (SVM) [34] for classification as no-logo or as belonging to a specific logo class. A multiclass one-vs-all linear SVM is employed with regularization hyperparameter $C = 1$.

5.6 Transformation Pursuit

Since the Flickr-Logo32 dataset has been collected to evaluate SIFT-like recognition algorithms [115] the training set contains only few examples for each logo (see Table 5.3). To handle the large extrinsic variability of the dataset and to prevent the learning algorithm to overfit, the the training set has been significantly increased following Transformation Pursuit [106]. For each region proposal extracted from images which overlaps with the groundtruth annotation a set of predefined image transformations is applied. In particular the applied tranformations include: translation, scale, shear on the y axis and shear on the x axis. With this set we take into account also rotation transformation which is a combination of the two shear transform. By applying only the two extrema values of each of the complete set of geometric transformations the number of examples can be increased by a factor of ~ 250 . In figure 5.7 is depicted a subset of the geometric transformations applied.



Fig. 5.7 Representative subset of geometric transformations applied to an extracted region proposal. The original image is in the lower-right corner.

5.7 Experimental Setup and Results

Experiments were performed on the publicly-available FlickrLogos-32 dataset [115]. This is a collection of photos showing 32 different logo brands, and is meant for the

Logo Recognition

evaluation of logo retrieval and multi-class logo detection/recognition systems on real-world images. All logos have an approximately planar or cylindrical surface. The whole dataset is split into three disjoint subsets P_1 , P_2 , and P_3 as reported in Table 5.3, each containing images of all 32 classes. The system was trained on FlickrLogos-32 P_1 set,

Table 5.3 FlickrLogos-32 dataset partitions

Partition	Description	Images	Total
P_1 (training set)	Single logo images, clean background	10 per class	320 images
P_2 (validation set)	Images showing at least a single logo Non-logo images	30 per class 3000	3960 images
P_3 (test set)	Images showing at least a single logo Non-logo images	30 per class 3000	3960 images

and validated on P_2 for hyperparameters selection with a target precision of 98% (for better comparison with the state of the art [115, 113, 114]). Finally, it was tested on P_3 with the selected hyperparameters.

Figure 5.8 shows the performance level obtained by the proposed method in terms of precision and recall, on validation set and test set. For the test set both performance with and without Query Expansion are reported, showing the significant gain obtained with this step.

Table 5.4 reports a comparison with other state of the art approaches. [115] and [114] are based on the bundling of neighboring SIFT-based visual words into unique signatures. [113] uses a statistical model for identifying incorrect detections output by keypoint-based matching algorithms.

Results show how even though the underlying CNN was trained for recognition in a different domain, it is still able to achieve near-state-of-the-art performance.

5.7.1 Robustness to Image Distortions

The robustness of the proposed method has been tested with respect to three different kinds of image distortion: blur, noise and lossy compression. Table 5.5 outline the strength of each transformation and Figure 5.9 shows an example of images from Flickr-Logos32 dataset with applied image distortions. Results showed that noise is the most affecting one, while lossy compression produce little to no performance loss.

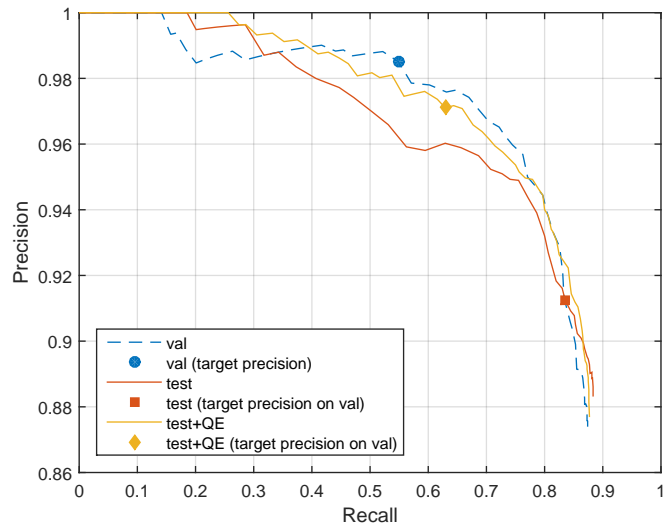


Fig. 5.8 Precision-Recall curve on validation set, test set, and test set with Query Expansion. Selected points are obtained by setting target precision at 98% in validation set.

Table 5.4 Performance comparison with other approaches

Method	Precision	Recall
Romberg et. al [115]	0.98	0.61
Revaud et. al [113]	≥ 0.98	0.73
Romberg et. al [114]	0.999	0.832
Proposed method	0.91	0.84
Proposed method + QE	0.97	0.63

Table 5.5 Types of distortions applied to the images of the FlickrLogos-32 dataset.

Type	Amount
Gaussian Blur	Filter Size 10px
Gaussian Blur	Filter Size 20px
JPEG Compression	Quality 20%
JPEG Compression	Quality 10%
Gaussian Noise	$\sigma^2 = 0.005$
Gaussian Noise	$\sigma^2 = 0.02$



Fig. 5.9 Types of distortions applied to the images of the FlickrLogos-32 dataset.

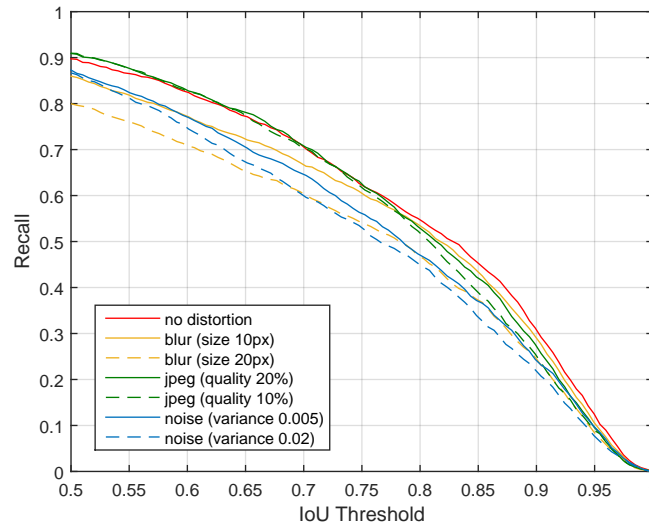


Fig. 5.10 Recall versus IoU threshold for 6 different types of image distortion on the Flickr-Logos dataset. Only the best overlapping bounding-box for each groundtruth annotation is considered.

5.7.2 Selective Search Evaluation

Figure 5.10 shows values of the Recall versus IoU threshold for the original and distorted images.

Image distortion has a low impact on the overall quality of the extracted Object Proposals. Blur has the biggest impact especially for low levels of IoU but in the worst case performance dropped by 10% only. On the other hand, jpeg compression seems to have a very low impact on the Object Proposals quality even at high levels of compression. Our tests confirm the results obtained by Hosang et al. in [60] which found the Selective Search to be one of the most robust Object Proposals algorithms.

In order to test the complete recognition pipeline on the distorted images, we augmented the training set by a factor of six. For each original image we add six deformed images, each with a single distortion applied. The magnitude of each distortion, shown in table 5.5, is the same for train and test. We run the recognition pipeline on images of the Flickr-logo test set modified with a single distortion at a time. This controlled environment makes it possible to check the impact of every single

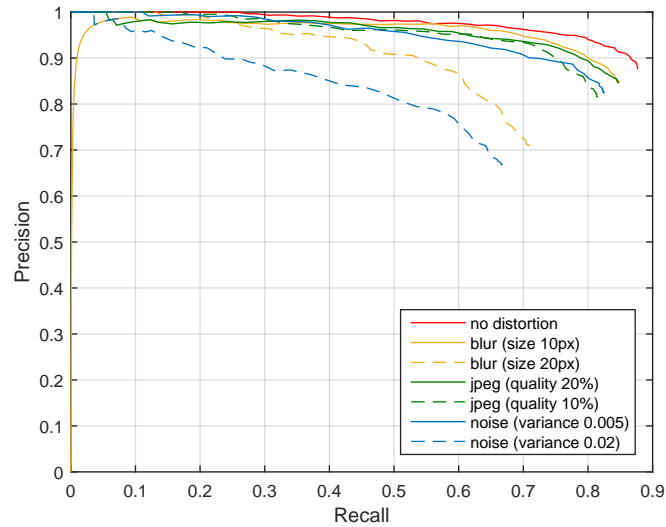


Fig. 5.11 Precision-Recall curves on the Flickr-logo test dataset. Different image distortions have been applied to obtain different curves.

distortion without masking effects.

Figure 5.11 shows the results of the performed tests. Gaussian noise and blur with high magnitude ($\sigma^2 = 0.02$ and $size = 20px$) have the highest impact on the overall results. The results on the complete recognition pipeline reflect those on the Selective Search part: the same types of distortions having the highest impact on the overall performance affect also the Recall measure of the Selective Search evaluation. This clue leads us to consider the quality of the Object Proposals as one of the most important aspects to care about in our recognition pipeline.

5.8 Improved pipeline

In this section is presented an improved processing pipeline for logo recognition. For each image, different object proposals are generated. These proposal are then cropped to a common size to match the input dimensions of the neural network and are propagated through a CNN specifically trained for logo recognition. There is an important difference with respect to the pipeline described in Section 5.8. Instead of using a pre-trained CNN as feature extractor with a linear classifier on top, the improved pipeline make use of an end-to-end trained neural network. This enhance the system accuracy and reduces the processing time since a smaller neural network can be employed. The proposed classification pipeline is illustrated in Figure 5.12.

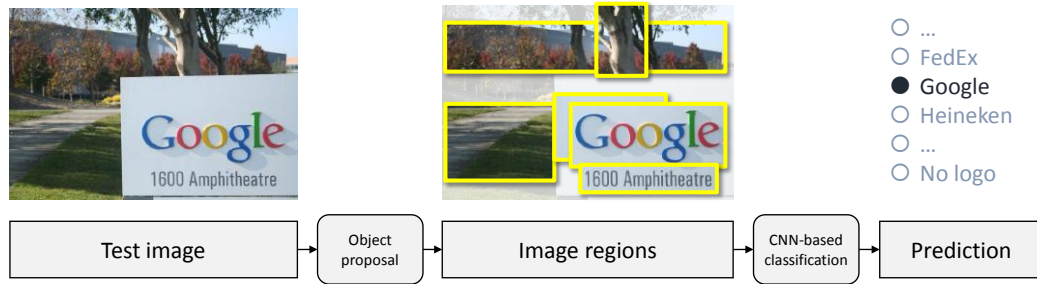


Fig. 5.12 Simplified logo classification pipeline

In order to have performance as high as possible within this pipeline, we use an object proposal that is highly recall-oriented. For this reason, the CNN classifier should be designed and trained to take into account that the logo regions proposed may contain many false positives or only parts of actual logos. To address these problems we propose here a training framework and investigate the influence on the final recognition performance of different implementation choices.

In more detail, the training framework is reported in Figure 5.13. The training data preparation is composed by two main parts:

- **Precise ground-truth logo annotations:** Given a set of training images and associated ground-truth specifying logo position and class, logo regions are first cropped and annotated with the ground-truth class. These regions are rectangular crops that completely contain logos but, due to the perspective of the image or the logo particular shape, may also contain part of the background.
- **Object-proposal logo annotations:** Regions that contains a logo should be automatically localized. To accomplish this, an object proposal algorithm is employed in the whole pipeline as shown in Figure 5.12. This algorithm is not applied only to the test images, but it is also run on the training images to extract regions that are more likely to contain a logo. Details about the particular algorithm used are given in the next section. Each object proposal in the training images is then labeled on the basis of its content: if it overlaps with a ground-truth logo region, it is annotated with the corresponding class and with the Intersection-over-Union (IoU) overlap ratio, otherwise it is labeled as background.

Within this training framework are investigated both the use of the precise ground-truth logo annotations alone or coupled with the object-proposal logo annotations. All positive instances, i.e. labeled logos and eventually object proposals that overlap with

them by a significant amount (i.e. $\text{IoU} \geq 0.5$), are used to train a Convolutional Neural Network whose architecture is given below. Different training choices are investigated within this framework in Figure 5.13:

- **Class balancing:** The logo classes are balanced by replicating the examples of classes with lower cardinality. Two different strategies are implemented: epoch-balancing, where classes are balanced in each training epoch, and batch-balancing, where classes are balanced in each training batch. The hypothesis is that this should prevent a classification bias of the CNN.
- **Data augmentation:** Training examples are augmented in number by generating random shifts of logo regions. The hypothesis is that this should make the CNN more robust to inaccurate logo localization at test time.
- **Contrast normalization:** Images are contrast-normalized by subtracting the mean and dividing by the standard deviation, which are extracted from the whole training set. The hypothesis is that this should make the CNN more robust to changes in the lighting and imaging conditions.
- **Sample weighting:** Positive instances are weighted on the basis of their overlap with ground-truth logo regions. The hypothesis is that this should make the CNN more confident on proposals highly overlapping with the ground truth logos.
- **Background class:** A background class is considered together with the logo classes. Background examples are not randomly selected, but are composed by the candidate regions generated by the object proposal algorithm on training images and that do not overlap with any logo. The hypothesis is that this should make the CNN more precise in discriminating logos and background class.

The actual contribution to the performance of each training choice considered will be discussed in Section 5.7.

After the CNN is trained, a threshold is learned on top of the CNN predictions. If the CNN prediction with the highest confidence is below this threshold, the candidate region is labeled as not being a logo, otherwise CNN prediction is left unchanged.

The testing framework is reported in Figure 5.14. Given a test image, Object Proposals are extracted with the same algorithm used for training. Contrast-normalization is then performed over each proposal (if enabled at training time), and feed them to the CNN. The CNN predictions on the proposals are max-pooled and the class identified with highest confidence (eventually including the background class) is selected. If

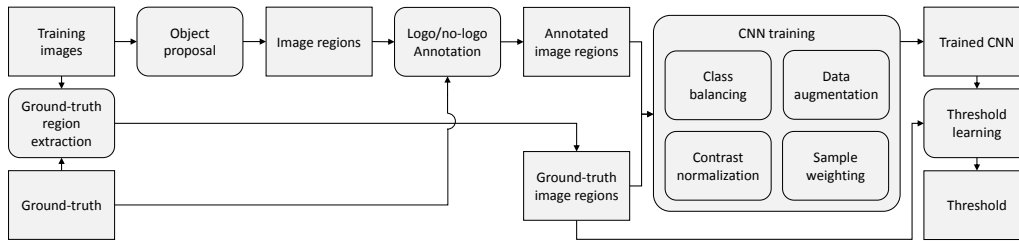


Fig. 5.13 Logo recognition training framework.

the CNN confidence for a logo class is above the threshold that has been learned in training, the corresponding logo class is assigned to the image, otherwise the image is labeled as not containing any logo.

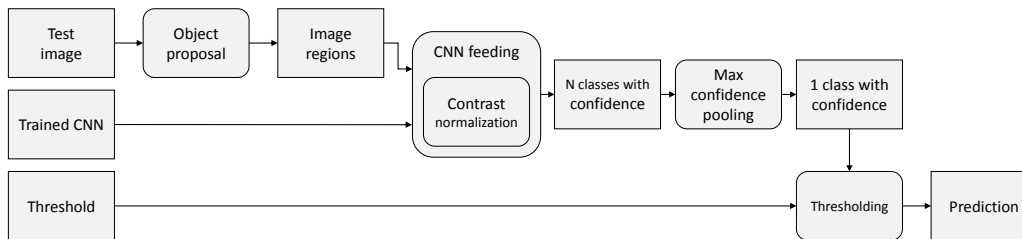


Fig. 5.14 Logo recognition testing framework.

5.8.1 Network Architecture

The architecture used for the experiments in the following sections is a tiny deep neural network. The reason to chose a tiny network is because it is fast at test time and it can be trained on cheap GPUs in very short time. It also allows to train the network without using any form of regularization like dropout [126], dropconnect [143], etc. decreasing even more the time needed for training and validating the network.

The same network structure was used by Krizhevsky in [76] on the CIFAR-10 dataset, where it was proven to be an high-performance network for the task of object recognition on tiny RGB images. It has three convolutional layers interleaved by ReLU nonlinearities and Pooling layers. All the pooling layers make the data dimensions halve after every Pooling block. The last part of the network (farthest from the input) consists in two Fully-connected layers with a final Softmax classifier. The whole net structure is presented in Table 5.6.

To give an idea of the network size, the proposed network has 1.5×10^5 parameters whereas AlexNet (used in [40]) and GoogLeNet (a similar structure is used in [61])

Table 5.6 Convolutional Neural Network Architecture

Layers	
1	Conv 32 filters of 5x5
2	Pool (max) with stride 2
3	Relu
4	Conv 32 filters of 5x5
5	Relu
6	Pool (average) with stride 2
7	Conv 64 filters of 5x5
8	Relu
9	Pool (average) with stride 2
10	Fully Connected of size 64
11	Fully Connected of size 33
12	Softmax

have respectively 6×10^7 and 1.3×10^7 parameters. Therefore the proposed network is less likely to overfit, even when the size of the training set is not large.

5.9 Experimental Setup and Results

Experiments are performed considering the different training choices described in Section 5.8. These include class balancing, data augmentation, image contrast normalization, sample weighting, addition of a background class, and addition of positive examples actually generated by the object proposal algorithm.

Each change to the training procedure is introduced one at a time, in order to assess its individual contribution, and the corresponding value is underlined in Table 5.7 for better readability. All these configurations are trained using real data augmentation, i.e. with the extended Logos-32plus dataset in addition to FlickrLogos-32 training and validation sets. Results are reported in Table 5.7 in terms of both F1-measure and Accuracy on FlickrLogos-32 test set. With reference to Figure 5.14, the threshold on CNN predictions is automatically chosen to maximize the accuracy on FlickrLogos-32 training and validation sets. The best configuration is then compared to other state of the art methods in Table 5.8. As further investigation the contribution given from real data augmentation is quantified by training the same solution on the original FlickrLogos-32 training set only. Finally, the impact of the object proposal algorithm

Logo Recognition

Table 5.7 Experimental results showing the impact of the different training choices described in Section 5.8 on the final classification. Results are reported in terms of Precision, Recall, F1-measure and Accuracy.

Train. Con-fig.	BG class	BBs	Data Augm.	Class bal.	Contr. norm.	Sample weight	Prec.	Rec.	F1	Acc.
I	No	GT	No	No	No	No	0.370	0.370	0.370	0.096
II	<u>Yes</u>	GT	No	No	No	No	0.713	0.665	0.688	0.620
III	Yes	<u>GT+OP</u>	No	No	No	No	0.816	0.787	0.801	0.744
IV	Yes	GT+OP	<u>Yes</u>	No	No	No	0.987	0.858	0.918	0.953
V	Yes	GT+OP	Yes	<u>Epoch</u>	No	No	0.986	0.865	0.922	0.956
VI	Yes	GT+OP	Yes	<u>Batch</u>	No	No	0.980	0.833	0.901	0.945
VII	Yes	GT+OP	Yes	Epoch	Yes	No	0.989	0.906	0.946	0.958
VIII	Yes	GT+OP	Yes	Epoch	Yes	<u>Yes</u>	0.984	0.875	0.926	0.951
IX	Yes	GT+OP	Yes	<u>Batch</u>	<u>Yes</u>	No	0.984	0.887	0.933	0.955
X	Yes	GT+OP	Yes	Batch	Yes	<u>Yes</u>	0.989	0.866	0.923	0.955

Legend to Table 5.7

Train. Config.	Identifier of the configuration used for training
BG class	Background class (no-logo examples) included in training
BBs	Bounding Boxes used as training examples
	GT Precise ground-truth logo annotations
	GT+OP Precise ground-truth and Object-proposal logo annotations
Data Augm.	Data Augmentation (translation)
Class bal.	Class balancing to account for different cardinalities
	Epoch Classes are balanced in each epoch
	Batch Classes are balanced in each batch as well
Contr. norm.	Pre-processing of training examples with contrast normalization
Sample weight	Weighting examples based on overlap between OP and GT

to the overall performance is assessed. To do this, instead of relying on the object proposal only, all the ground truth locations are added to the test set.

From the results reported in Table 5.7 it is possible to see that with respect to a straightforward application of deep learning to the logo recognition task (i.e. Training Configuration I, *TC-I*), the different training choices considered are able to give a large increase in performance:

- The first jump in performance is obtained by including the background (i.e. no-logo examples) as a new class in training. Results are identified as *TC-II* and show an improvement in F1-measure and accuracy of 31.8% and 52.4% with respect to *TC-I*.
- A second jump is obtained by including object proposals coming from Selective Search as additional training examples. This configuration is named *TC-III*

and improves the F1-measure and accuracy by 11.3% and 12.4% with respect to *TC-II*.

- A third jump in performance is obtained by augmenting the cardinality of object proposals coming from Selective Search by perturbing them with random translations (i.e. synthetic data augmentation). This configuration is named *TC-IV* and improves the F1-measure and accuracy by 11.7% and 20.9% with respect to *TC-III*.
- A further, smaller, improvement in performance is obtained by considering class balancing to account for different cardinalities, with “Epoch” balancing giving consistently better performance than the “Batch” counterpart (named *TC-V* and *TC-VI* respectively). In particular, *TC-V* improves the F1-measure and accuracy by 0.4% and 0.3% with respect to *TC-IV*.
- Contrast normalization brings a further little but consistent improvement, with *TC-VII* improving the F1-measure and accuracy by 2.4% and 0.2% with respect to *TC-V*.
- Sample weighting instead (adopted in *TC-VIII* and *TC-X*), which consists in weighting training examples according to the degree of overlap between the object proposal and ground truth regions, results in lowering the final performance of the method.

The best configuration (i.e. *TC-VII*) trained on the extended training set is highlighted in bold in Table 5.7 and compared with the state of the art in Table 5.8. Performances of the other methods are taken from the respective papers and thus for some of them some performance measures are missing. From the results reported it is possible to see that the proposed solution is able to improve the F1-measure with respect to the best method in the state of the art by 3.8%, and the accuracy by 1.7%. It is worth to underline that the best results for the two metrics were obtained by different methods in the state of the art, i.e. by Romberg et. al [114] and BoW SIFT [114] respectively.

As a further comparison, the results obtained by the proposed solution are reported using only FlickrLogos-32 for training and keeping all the other training choices unchanged. This results in a drop in F1-measure by 14.7% and by 4.8% in accuracy, giving an idea of the benefit of real data augmentation with respect to a purely synthetic one [40]. As a final analysis, to understand if the major source of error in our method is the Selective Search module that is unable to have a high recall or if its the CNN itself that mispredicts the logo class, an additional test is performed by adding the

Logo Recognition



Fig. 5.15 Wrongly labeled logos ordered by confidence. Highest confidence prediction is top-left. Images resolution is 32×32 pixels, i.e. the same used to feed the CNN. The first and third rows are the wrongly labeled logos, the second and the fourth rows represent the nearest example in the training set (using the last network layer activations before the softmax as feature vector).

actual logo ground truth region to the object proposals. This increases the F1-measure by 0.6% and the accuracy by 0.2% indicating that the major source of error in this method is the CNN itself. Some examples of wrongly labeled candidate logo regions are reported in Figure 6.7. Candidates are generated by the object proposal and they have a IoU larger than 0.5 with the corresponding ground truth. The first and the third row depict the wrongly recognized regions labeled with their actual class, while the second and fourth one depict the nearest example in the training set using as features the activations of the last network layer before the softmax. Images are reported with the same resolution used to feed the CNN, i.e. 32×32 pixels.

5.10 Timings

Table 5.9 shows the timings for the whole recognition procedure at test time. Experiments are performed on the same computer (Intel i7 3.40 GHz - 16 GB RAM) averaging the timings of 100 runs on different images.

Two different solutions are compared: the use of CPU or GPU (GeForce GTX 650) for the classification step.

The proposals extraction step runs always on CPU. The preprocessing time include the resize of every patch to match the CNN input size, the contrast normalization

Table 5.8 Comparison of the best configuration in Table 5.7 with the methods in the state of the art. Last two rows are obtained by adding the groundtruth to the object proposal at test time. This results in a quantitative evaluation of the classification network bypassing errors from the object proposal algorithm.

Method	Train data	Precision	Recall	F1	Accuracy
BoW SIFT [114]	FL32	0.991	0.784	0.875	0.941
BoW SIFT + SP + SynQE [114]	FL32	0.994	0.826	0.902	N/A
Romberg et. al [115]	FL32	0.981	0.610	0.752	N/A
Revaud et. al [113]	FL32	≥ 0.980	0.726	0.834÷0.841	N/A
Romberg et. al [114]	FL32	0.999	0.832	0.908	N/A
Bianco et. al [19]	FL32	0.909	0.845	0.876	0.884
Bianco et. al + Q.Exp. [19]	FL32	0.971	0.629	0.763	0.904
Eggert et. al [40]	FL32	0.996	0.786	0.879	0.846
Oliveira et al. [102]	FL32	0.955	0.908	0.931	N/A
DeepLogo [61]	FL32	N/A	N/A	N/A	0.896
Ours (<i>TC-VII</i>)	FL32	0.976	0.676	0.799	0.910
Ours (<i>TC-VII</i>)	FL32, L32+	0.989	0.906	0.946	0.958
Ours (<i>TC-VII</i> , +GT)	FL32	0.968	0.755	0.848	0.917
Ours (<i>TC-VII</i> , +GT)	FL32, L32+	0.989	0.917	0.952	0.960

Table 5.9 Timings of the whole recognition pipeline running on CPU and GPU. Time decomposed on proposal, preprocessing and classification stages.

Device	Proposal	Preproc.	Classif.	Overall
CPU	1.24 s	0.93 s	0.71 s	2.91 s
GPU	1.24 s	2.12 s	0.36 s	3.74 s

Logo Recognition

(negligible processing time) and eventually the time to copy the data from CPU to GPU memory. In Table 5.9 it is possible to notice that the overhead caused by the CPU-GPU memory transfer makes the overall time of the GPU solution higher than that of the CPU solution. To this extent, in the future it might be interesting to evaluate a fully GPU-based pipeline, for example generating and pre-processing proposals according to [112].

Chapter 6

Painting Categorization

Research on digital analysis of paintings is gaining increasing attention due to the large quantities of visual artistic data [30, 91, 70], made available from art museums digitizing their collection for cultural heritage, and the need of automatic tools to organize and manage them. In this work, we approach the problem of categorizing a painting by automatically predicting its artist and style given solely the digital version of the painting itself [12]. Both these tasks are very challenging due to the large amount both inter- and intra-class variations, e.g. the different personal styles in the same art movement, or the same artist adhering to different schools in different periods in his/her production. Artist classification consists in automatically associate the painting to its painter. In this task factors such as stroke patterns, the color palette used, the scene composition, and the subject must be taken into account. Style classification consists in automatically categorize a painting into the school or art movement it belongs to. Art theorists define an artistic style as the combination of iconographic, technical and compositional features that give to a work its character [147]. Style categorization is complicated by the fact that styles may not remain pure but could be influenced by others.

In this Chapter a multiresolution approach to solve the tasks of artist, style and genre categorization is proposed. Two different crop strategies are adopted to gather clues from low-level texture details and, at the same time, exploit the coarse layout of the painting. The first is a particular random-crop strategy that permits to manage the tradeoff between accuracy and speed and the second is a smart extractor based on Spatial Transformer Networks [64]. The joint use of handcrafted along with neural features is experimented and the use of multitask classification and synthetic data augmentation are investigated as regularizers.

Experiments are performed on a new dataset suitable for multitask learning composed by 100K paintings. The proposed method is compared with the state-of-the-art on the challenging painting-91 dataset [70]. On both artist and style classification tasks the proposed approach improves the mean classification accuracy by 14.3% and 10.2% respectively, compared to the previous state-of-the-art models.

6.1 Related works

The problem of painter or style categorization has been faced using different techniques. Some existing approaches make use of traditional handcrafted features [30, 70] whereas more recent works rely on the use of deep networks [108, 107, 12, 130]. Peng et al. [108] use a multiresolution approach to exploit both small details and the overall image structure. A more sophisticated technique is used by [12] where the use of a deformable part model is adopted in order to combine low-level details and an holistic representation of the whole painting. Deep CNNs have been widely used as features extractors to solve different tasks [123, 22], Peng et al. [108] and Anwer et al. [12] rely on pretrained deep CNNs to deal with the small quantity of images of the Painting-91 dataset. Tan et al. [130] made different experiments by training a network from scratch or finetuning an existing network for the task of style and painter recognition. They adopted a network structure similar to the one used by Krizhevsky et al. [76]. Hentschel et al. [59] performed interesting experiments about the quantity of data needed to fine-tune the network by Krizhevsky et al. [76] for the task of style classification.

6.2 Proposed Approach

The scene composition and the subject depicted are important clues to recognize a particular author or a painting style. These elements need to be extracted from the whole painting. At the same time finer details, such as stroke patterns or the line styles, are also very good clues. Obviously a powerful discriminative model should consider both the coarse level and fine details. On the basis of these considerations the proposed approach adopts a multiresolution approach: first, a predefined number of squared "small" crops are extracted from the high-resolution image using strategies presented in the next sections. Then, the image is downsampled and another "large" crop is extracted from the low-resolution image (see Sec. 6.2.1). All the crops are then fed to the branches of a deep neural network that extracts the corresponding features.

The outputs of the branches are collected by a join layer and fed to a deep neural network that carries on the categorization process.

6.2.1 Input preprocessing

The first preprocessing step consists in normalizing the input image by subtracting the mean and dividing by the standard deviation of the pixel distribution of whole training set. This contrast normalization preprocessing is known to improve CNNs accuracy in different domains [20] by limiting the variability of the input range. The second step consists in a particular cropping strategy. Crops are taken at multiple resolutions to capture both fine details and coarse structures. Since paintings exhibit high variability in terms of aspect-ratios, the input image is resized such as the minimum side is 512 pixels and the aspect ratio is preserved. From the resulting image we extract two squared random crops of 227 pixels side. Then the image is further downsampled such as the minimum side is 256 pixels and another squared crop of 227 by 227 pixels is extracted. All the crops are squared, independently from the original aspect ratio of the input image. This is done to improve the computational efficiency allocating GPU memory blocks only once. Images and crops sizes has been chosen as a tradeoff to exploit fine details and to limit the computational burden accordingly to the size and quality of the original images. The coordinates of the crops inside the input image are randomly chosen with the only constraint that crops coming from the same scale do not overlap. The rationale behind this choice is that the salient details can be anywhere inside the painting, and the extraction of crops at random locations permits the implementation of a consensus strategy by simply processing the same input image several times. The consensus strategy consists in averaging the output of the last fully-connected layer for the multiple passes of the same image through the network, resulting in a feature vector that is then fed to the softmax layer to get the final prediction.

A deterministic crop extraction using Spatial Transformer Networks [64] is also investigated.

6.2.2 Spatial Transformer Network

Spatial Transformer Network (STN) has been introduced by Max Jadeberg et al. to explicitly model the spatial manipulation of data within the network. It is composed by three modules. The first is a *Localization Network* that takes the input feature map $U \in \mathbb{R}^{H \times W \times C}$ where H, W, C represent width, height and channels respectively. This

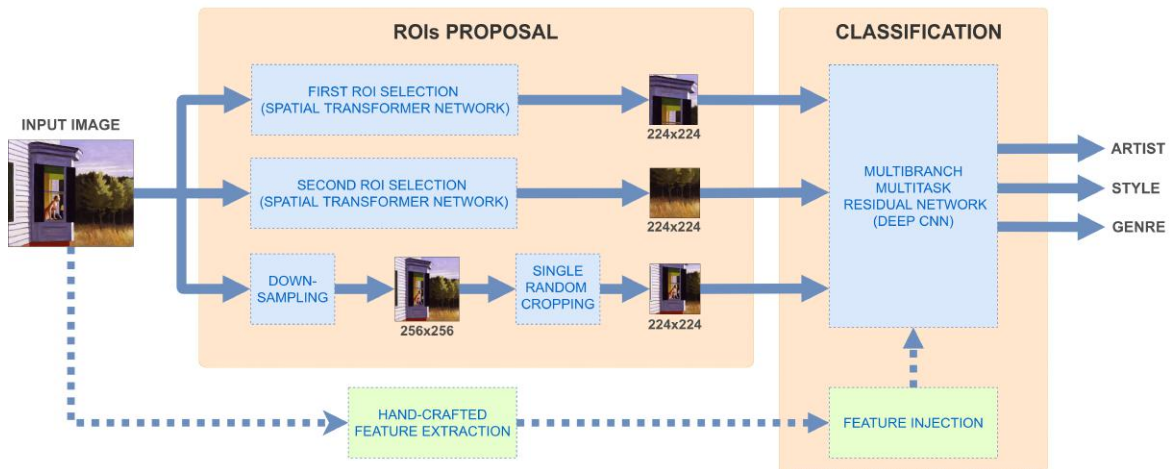


Fig. 6.1 Scheme of the Deep Multibranch Multitask Neural Network.

network outputs the parameters θ of the transformation to be applied to the feature map: $\theta = f_{loc}(U)$. The second is a *Parametrized Sampling Grid* module that takes as input the parameters from the Localization Network and produces a sampling grid. The third one is the *Bilinear Sampler* which is a differentiable bilinear interpolation layer that takes as input the feature map and the sampling grid and perform the actual spatial warping.

The type of transformations handled by the Sampling Grid layer is limited allowing only translation and scale in order to maintain geometric structure of paintings. With such configuration the Spatial Transformer Network becomes a smart crop extractor as a substitution to the random crop strategy. As Localization Network a ResNet-18 [58] is used with the same type of Residual Blocks used for the main network and described in Section 6.2.3.

6.2.3 Deep Network Structure

The structure of the proposed novel network is shown in Figure 6.1. It is composed of five modules: three branches to extract the low level structures of the painting crops, a join module to gather the output of the three branches and a classification module to make the prediction. Each branch is trained with crops from a specific scale, thus becoming specialized in processing texture patterns at that specific resolution. We decided to use only two scales since, in our preliminary experiments, the use of higher scales brought a slight improvement compared to the exponential increase of computational burden.

Table 6.1 Structure of the Multibranch Multitask Deep Neural Network.

Output Size	Layers		
	branch 1	branch 2	branch 3
112x112x64	Conv7	Conv7	Conv7
	BatchNorm	BatchNorm	BatchNorm
	ReLU	ReLU	ReLU
	MaxPool	MaxPool	MaxPool
56x56x256	3× ResBlock	3× ResBlock	3× ResBlock
56x56x768	Concatenation (channel dimension)		
28x28x512	ResBlock, stride 2		
	2× ResBlock		
14x14x1024	ResBlock, stride 2		
	5× ResBlock		
7x7x2048	ResBlock, stride 2		
	3× ResBlock		
1x1x2048	AvgPool		
Num. Classes	FC-1508	FC-125	FC-41

In the three branches and in the classification model the proposed deep network makes use of Residual Blocks which have been shown to be an effective architectural choice to build very deep networks [58] and tackle the problem of vanishing gradients by using shortcut connections. In particular, it makes use of "bottleneck" Residual Blocks, which allow the network architecture to be even deeper [58]. Each skip connection has four times the number of channels with respect to the internal elements of the block. This permits a large throughput of information among layers while maintaining a low computational complexity and low memory use inside each block. The Residual Block structure adopted is different from the one used by He et al. [58]: the Batch Normalization layer [62] is moved after the sum with the skip connection because, in preliminary experiments, the resulting configuration has shown better performances.

The Residual Block adopted is shown in Figure 6.2. In the proposed network (see Fig. 6.1) each of the three branches is composed by three Residual Blocks plus four layers near the input which perform the first processing (Convolution + BatchNorm[62] + ReLU[98]) and an initial downsampling (Max Pooling). The join module is a particular Residual Block which gathers the output of the three branches. It stacks the output features and then converts them to a smaller-dimensional feature space by compressing information along the channel dimension. The reason behind this operation is to make the computations feasible in the following layers by reducing the channel dimension of the output by a factor of three.

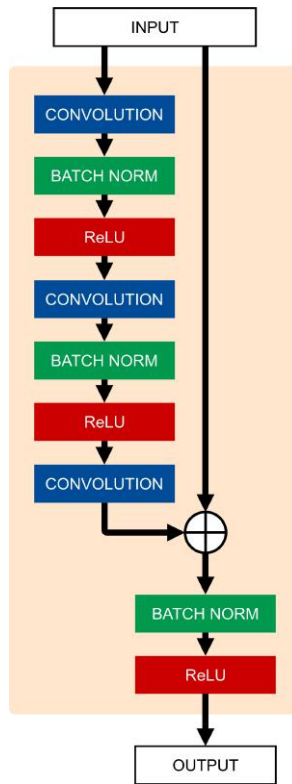


Fig. 6.2 The type of Residual Block used in the proposed Deep Neural Network

The classification module is composed by 13 Residual Blocks plus a Spatial Average Pooling layer, a Fully-connected layer and a Softmax layer that outputs the classes probabilities. While the Residual Blocks in the three branches do not include any downsampling operator, the classification module uses convolution operators with stride two to perform a spatial downsampling of the input. Every five blocks the input is spatially reduced by a factor of two. At the same time the number of channels is increased by the same amount. This leads to a gradual increasing of the receptive-fields of the network in the deeper layers and also favors more abstract representations of the input. In the final part of the classification module a fully-connected layer maps the output to the right number of classes depending on the task, respectively artist, style or genre categorization.

6.2.4 Hand-crafted feature injection

The joint use of handcrafted features along with learned neural features is investigated. To assess the improvement that handcrafted features could bring to the existing architecture some preliminary experiments have been performed. A linear classifier is

trained on top of each feature to classify for each of the three tasks. This experiment gives a first glance on the discriminative power of the considered features for the final classification tasks.

The second preliminary experiment performed is to use the trained linear classifier on top of handcrafted features to classify only the misclassified examples of the existing neural network architecture. While the first experiment gave an idea of the discriminative power of the features for the tasks, this second experiment can assess if a descriptor can help our current architecture for misclassified examples.

Considering the results of these preliminary experiments HOG is chosen as handcrafted descriptor to be added to the classification pipeline. Extracted features are fed directly before the last fully-connected layer.

6.3 Datasets

6.3.1 Painting91

The recognition pipeline is evaluated on Painting91 dataset [70] for artist and style classification tasks. The dataset consists of 4266 paintings of 91 painters. As train and test split those provided by the authors are used which are in both cases of artist and style classification nearly 50%. For the task of artist recognition, the whole dataset is used whereas for the task of style recognition only 2338 groundtruth are provided.

6.3.2 Wikipaintings-IVL Dataset

Wikipaintings-IVL is a new dataset for multitask painting classification collected by the author. Original images and annotations comes from Painter by Numbers Kaggle competition.¹ and have been downloaded mostly from wikiart.org plus some additional other sources. Wikipaintings-IVL dataset is composed by a large subset of the Painter-by-numbers dataset but with a different train-test split. The goal of Painter-by-numbers competition was to predict if a pair of images are artworks made by the same artist or not. Therefore some artists had paintings in both train and test splits while others were only in the testset. Wikipaintings-IVL is collected to evaluate algorithms on another task: given a new painting, predicting the author among a list of known painters. Moreover another goal is to solve a multitask classification problem (i.e predict painting's author, style and genre), therefore a subset of the original dataset

¹<https://www.kaggle.com/c/painter-by-numbers>

Painting Categorization



Fig. 6.3 Paintings from Wikipaintings-IVL dataset. Each row contains samples from a different style, i.e. from top to bottom: Impressionism, Baroque, Pop-art and Symbolism

is kept such that there are at least 10 images for each class. This condition should be consistent for each of the three tasks (author, style and genre). After this selection the dataset contains 99816 images with respect to 103250 of the original dataset. 70% belongs to the train set and 30% to the test set. The resulting dataset is composed by 1508 artists, 125 styles and 41 genres. Images are at different resolutions but in general not smaller than 512px per side. Table 6.2 reports a comparison between Painting-91 and Wikipaintings-IVL datasets. In particular Wikipaintings-IVL contains two orders of magnitude more the images of Painting-91. Moreover the number of painters is considerably larger, making Wikipaintings-IVL a very challenging dataset.

Table 6.2 Comparison between Paintings-91 and Wikipaintings-IVL datasets. The latter contains a larger number of images and classes.

Dataset	Paintings-91	Wikipaintings-IVL
Images	4266	99816
Painters	91	1508
Styles	12	125
Genre	n/a	41

6.3.3 Synthetic Data Augmentation

The Wikipaintings-IVL dataset contains artworks by 1508 painters. For 30 artists like picasso, gaugin, cezanne etc. there are more than 300 paintings whereas for some painters there are less than 20 paintings. To cope with this long tail problem a synthetic data augmentation technique based on Neural Style Transfer [47] is investigated. Gatys

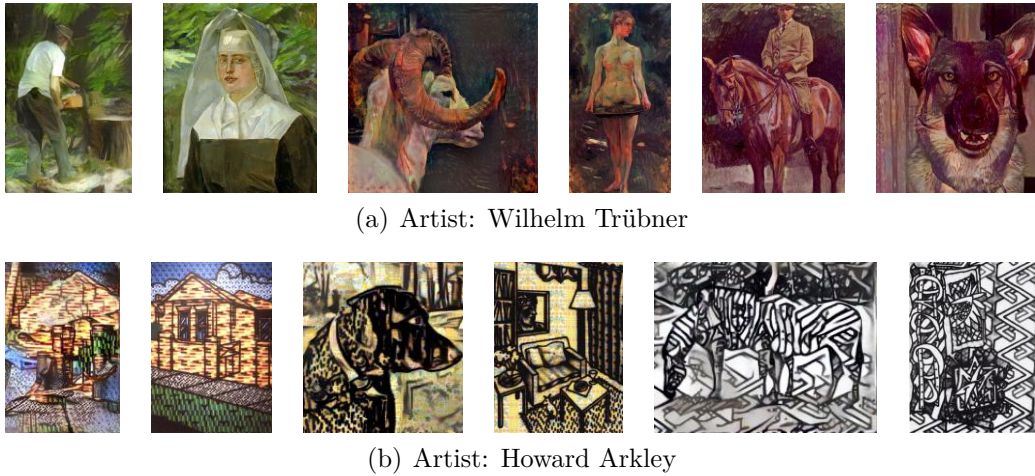


Fig. 6.4 Examples of real and synthetic paintings. For each row, paintings 1, 3 and 5 are synthetic whereas the others are real paintings used as source for style transfer. Content images are from ILSVRC validation set.

et al. in their work generate artistic images of high perceptual quality by matching statistical properties of Deep Neural Networks activations. The system is composed by three siblings neural networks, i.e. pretrained VGG [124]. The backpropagation algorithm is used to shape the output image in order to match patterns of the input content image and the input style image. The resulting image contains the coarse structure of the input content image and at the same time the textures of the input style image resulting in a new artwork. For an in-depth discussion about the generative algorithm used to augment the painting dataset see Appendix A which is entirely dedicated to the neural style transfer architecture.

Images from the validation set of the ILSVRC dataset [121] are used as content images to produce new paintings and use as style images the real paintings from a specific artist. Figure 6.4 shows some real and generated paintings of Wilhelm Trübner. Best perceptual results are obtained with pictorial styles characterized by the use of visible strokes.

6.4 Experiments

6.4.1 Training

To cope with the small amount of training data we exploited some data augmentation techniques:

Painting Categorization

- Color Jitter. It consists in randomly modifying contrast, brightness and saturation of the input image independently.
- Lighting noise. It is a pixelwise transform based on the eigenvalues of the RGB pixel distribution of the dataset. It has been introduced by Krizhevsky et al. [76].
- Gaussian Blur. It consists in applying a blur filter with fixed σ to random images chosen with probability 0.5.
- Geometric transforms. It includes small changes in scale and aspect-ratio of the input image.

6.4.2 Results on Painting-91 dataset

Training procedure on Painting-91 was carried out in two phases. First the deep network architecture is pretrained on the Wikipaintings-IVL dataset. Then it is finetuned two times (one for each of the two tasks) on the Painting91 dataset, substituting the last fully connected layer with a new one that matched the number of classes needed for each task.

As explained in the subsection 6.2.1 the proposed network architecture exploits random crops. Therefore if the same input is processed several times by the same network, the final prediction vectors can be averaged before being fed to the last softmax layer. In Table 6.3 the performance in terms of accuracy at different number of passes are reported. Results are averaged over ten independent runs. The biggest improvement is obtained by exploiting two passes with respect to the single one. The best performance are obtained using four passes.

Table 6.3 Accuracy vs Number of Passes through the Network. Each value represents the average of 10 runs.

Passes	1	2	4	8
Artist	77.5	78.1	78.5	78.3
Style	83.6	84.1	84.4	84.3

In Table 6.4 are reported the performances of the proposed method with respect to the state-of-the-art on the Paintings-91 dataset. Concerning the proposed method, the average accuracy over ten independent runs together with the minimum and maximum values are reported. Considering the average performance, the proposed

Table 6.4 Comparison with the state of the art. Average classification rates on the Paintings-91 dataset for the tasks of Artist and Style recognition.

Method	Artist	Style
VGG-16 FC [124]	51.7	67.2
MF [70]	53.1	62.2
CL-CNN [107]	56.4	69.2
MS-MCNN [108]	58.1	71.0
MOP [51]	59.7	68.8
Holistic [32]	61.8	70.1
Holistic + Part Based [12]	64.5	74.8
Proposed Single Branch	74.9	83.8
Proposed Multi-branch (average performance among 10 runs)	78.5	84.4

method outperforms the best method in literature by 14.0% and 9.6% on the task of artist and style categorization respectively.

Figure 6.5 shows the confusion matrix for the style recognition task. The highest classification errors are between the Neo-Classical, Baroque and Renaissance classes. This seems to agree with styles’ contaminations and influences as studied by art historians. For example Caravaggio paintings are classified as Baroque in Paintings-91 groundtruth. Actually he lived at the end of the Renaissance era, having a great influence on future Baroque painters.

Figure 6.6 shows the confusion matrix for the task of artist recognition. The highest error rates are between Memling and Van Eyck (27%), and Zurbaran and Vermeer (30%). Memling and Van Eyck are contemporaneous and both belonging to the Dutch and Flemish Renaissance, while Zurbaran and Vermeer are coeval painters, both belonging to the Baroque movement. To be able to actually discriminate between the last two painters, the network should be aware that Vermeer paintings are usually about indoor every-day life scenes whereas Zurbaran mostly painted religious subjects.

6.4.3 Multitask and Spatial Transformer Network

In Table 6.5 are reported the results of different experiments performed on Wikipaintings-IVL dataset. A joint multitask training on all tasks gives a big boost on style accuracy and a small decrease in performance on genre. The same happens with the injection of HOG features. We suppose that artist and style are much more correlated tasks, thus the training can benefit more from a joint loss optimization. The use of Spatial Transformer Networks improves the performances on all tasks showing the contribution of the smart crop extraction strategy.

Painting Categorization

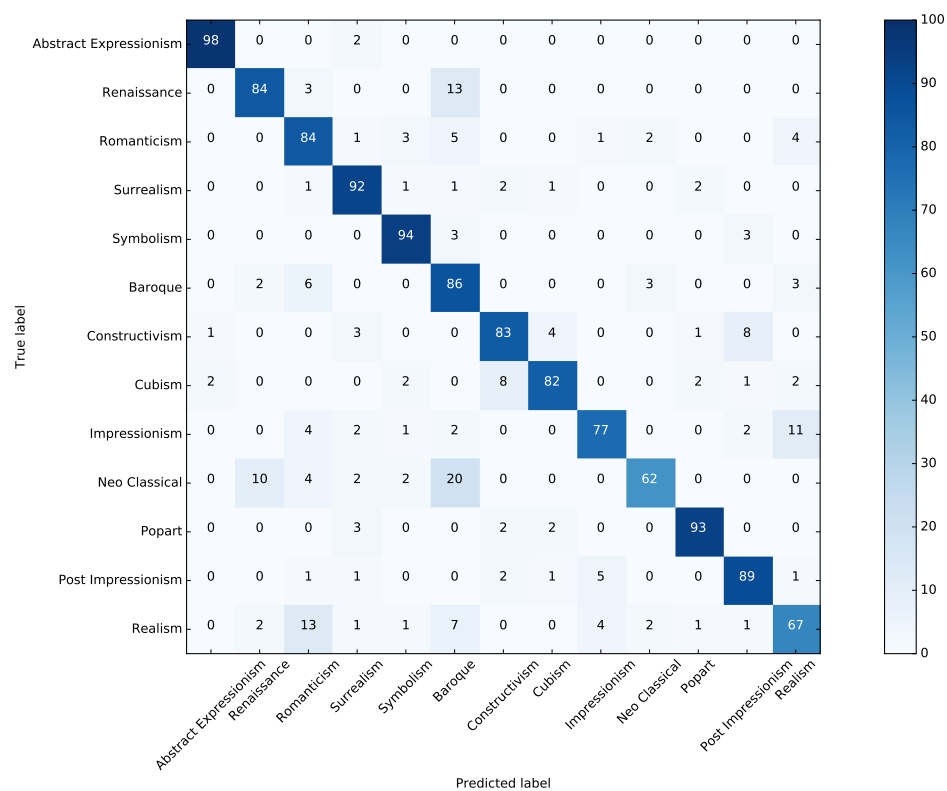


Fig. 6.5 Confusion matrix for the task of style recognition. The highest error rates are between Neo-Classical paintings, Baroque and Renaissance.

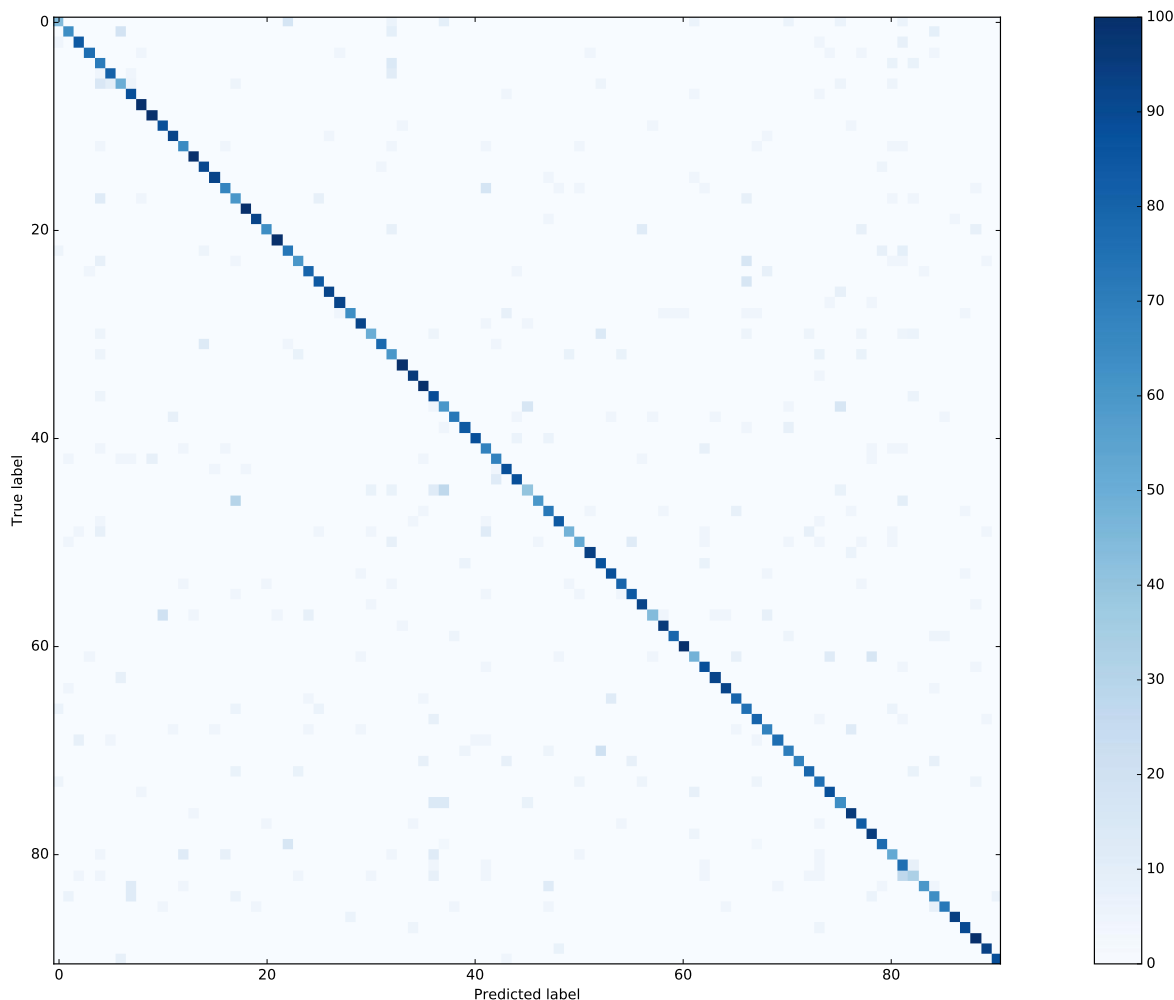


Fig. 6.6 Confusion matrix for the task of artist recognition. The highest error rates are between Zurbaran and Vermeer, Memling and Van Eyck. These painters are coeval and belongs to the same artistic movement.

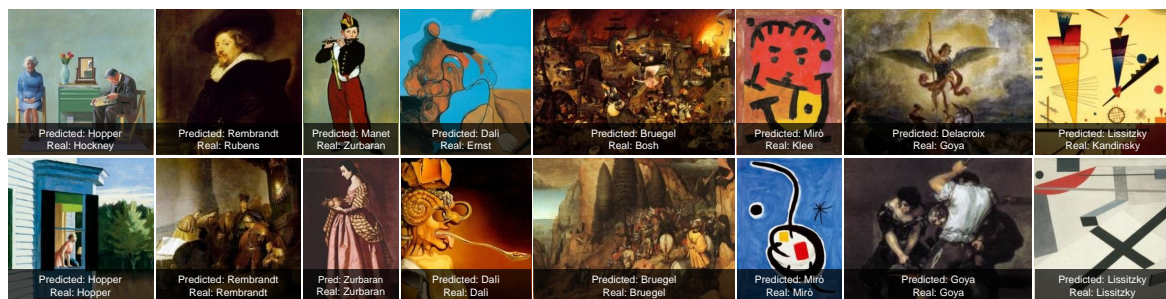


Fig. 6.7 Top row: highest scored errors for the task of painters classification. Bottom row: for each of the predicted painters, we report the correctly classified example with the highest score.

Painting Categorization

Table 6.5 Classification accuracy for different tasks on Wikipaintings-IVL dataset. Different models exploiting multibranch, multitask and Spatial Transformer Networks

Model	Artist	Style	Genre	Average
Multibranch	53.1	51.5	64.3	56.3
Multibranch multitask	53.3	55.4	63.0	57.2
STN multitask	56.1	57.0	64.1	59.1

6.4.4 Synthetic Data Augmentation

50 artists with very few examples in the dataset, i.e. 10 are considered for the experiments. The number of their paintings is synthetically increased by 15x with the method exposed in Section 6.3.3. Then two different experiments are performed. The first consists in adding synthetic paintings to the dataset assigning the same label as the style painting used as input for synthesis. The second experiment consists in assigning different labels for real and synthetic (fake) paintings and thus increasing the number of classes. The network has one more task now: discriminate fake from real paintings. The hypothesis behind this choice is that this new task could act as a regularizer. Table 6.6 shows the the results of both experiments. The baseline experiments is the architecture with Spatial Transformer Network but without multitask since the dataset is augmented only with respect to the painters. The first experiment gives a slight improvement of 0.2% whereas the second experiment gives a more consistent boost to the system accuracy, i.e. 2.2%. The multitask experiment is also included for comparison which results in a stronger regularizer with respect to the proposed data augmentation technique.

Table 6.6 Accuracy measures obtained using synthetic data either as a form of data augmentation or regularization.

Model	Artist prediction (accuracy)
NO multitask (baseline)	52.7
NO multitask + augmentation same classes	52.9
NO multitask + augmentation + new classes	54.9
Multitask	56.1

6.4.5 Handcrafted descriptors

A huge variety of features have been proposed in literature for describing the 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 CNNs.

In this thesis several descriptors from the state of the art have been adopted, by taking a few representative descriptors for each of the approaches mentioned above. Several descriptors have been applied to both color and gray-scale images, where the gray-scale image is defined as the luminance of the image and is obtained by using the standard formula: $y = 0.299R + 0.587G + 0.114B$:

- 256-dimensional grey-scale histogram (Hist y) [100];
- 768-dimensional RGB marginal histograms (Hist RGB) [110];
- 144-dimensional Colour and Edge Directivity Descriptor (CEDD) features. This descriptor uses a fuzzy version of the five digital filters proposed by the MPEG-7 Edge Histogram Descriptor (EHD), forming 6 texture areas. CEDD uses 2 fuzzy systems that map the colours of the image in a 24-colour custom palette;
- 8-dimensional Dual Tree Complex Wavelet Transform (DT-CWT) features obtained considering four scales, mean and standard deviation, and three colour channels (DT-CWT and DT-CWT L) [24, 14];
- 512-dimensional Gist features obtained considering eight orientations and four scales for each channel (Gist RGB) [101];
- 32-dimensional Gabor features composed of mean and standard deviation of six orientations extracted at four frequencies for each colour channel (Gabor L and Gabor RGB) [24, 23];
- 580-dimensional Histogram of Oriented Gradients feature vector [69]. Nine histograms with nine bins are concatenated to achieve the final feature vector (HoG);
- 18-dimensional Local Binary Patterns (LBP) feature vector for each channel. We considered LBP applied to grey images and to colour images represented in RGB [86]. We selected the LBP with a circular neighbourhood of radius 2 and 16 elements, and 18 uniform and rotation invariant patterns (LBP y and LBP RGB);
- 10-dimensional feature vector composed of normalized *chromaticity moments* as defined in [104];

Painting Categorization

- 499-dimensional Local Color Contrast feature vector. It is obtained by concatenating the LBP on the gray images with a quantized measure of color contrast [35] (LBP-LCC);
- SIFT (Bag of Words): 1024-dimensional BoVW of Scale Invariant Feature Transform (SIFT) descriptors extracted from regions at given key points chosen using the SIFT detector (SIFT). We built a codebook of 1024 visual words by exploiting images from external sources [65].

Figure 6.8 shows the percentage of accuracy for each handcrafted feature tested and for each task.

As expected performances for the task of artist prediction are very bad. This is the most difficult of the three tasks due to the large set of classes i.e. 1508. On style and genre prediction some descriptors show an accuracy over 4%. In particular the best features for style prediction are HOG and Gabor y , both grayscale descriptors, whereas for genre prediction genre classification the best descriptors are GIST color and chromaticity moments which relies both strongly on color information.

Figure 6.9 contains a stacked bar graph. Each bar represent the cumulative contribution for all of the three tasks. From this graph are clearly visible the features that gives a higher overall contribution: HOG, Gabor y , Chromaticity Moments and dt cwt. Since the best improvement is given by HOG, it has been chosen as handcrafted descriptor to be injected in the Neural Network architecture. Table 6.7 shows a comparison on the use of the HOG descriptor.

Table 6.7 Classification accuracy for different tasks on Wikipaintings-IVL dataset. Performances of the STN multitask network with and without HOG features injection.

Model	Artist	Style	Genre	Average
STN multitask	56.1	57.0	64.1	59.1
STN multitask + HOG	56.5	57.2	63.6	59.1

6.5 Similarity search

Deep Neural Networks learn very complex feature hierarchies that are usually difficult visualize. Understand the exact factors that induce a particular decision by the network is not a trivial task. In the last years different methods have been investigated to try to understand more in depth this process, e.g. [87]. One possibility is for example

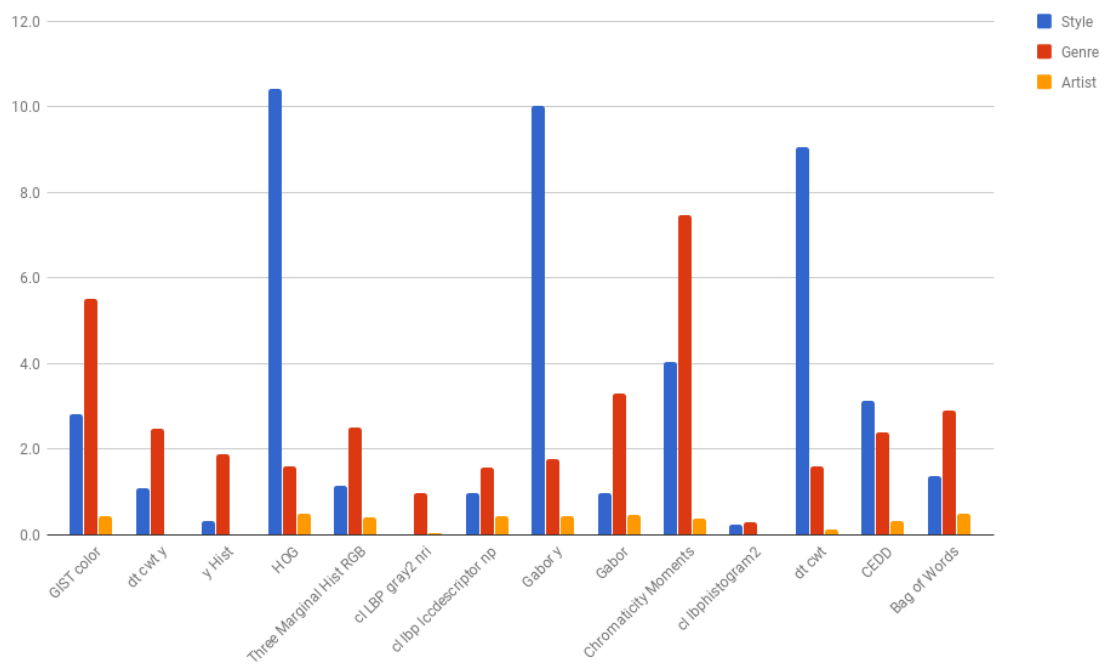


Fig. 6.8 Classification accuracy (percentage). Handcrafted features plus a linear classifier to solve the three classification tasks. *cl LBP gray2 nri* stands for LBP-L, *cl lbp lccdescriptor np* stands for LBP-LCC, *cl lbp histogram* stands for LBP-RGB

Painting Categorization

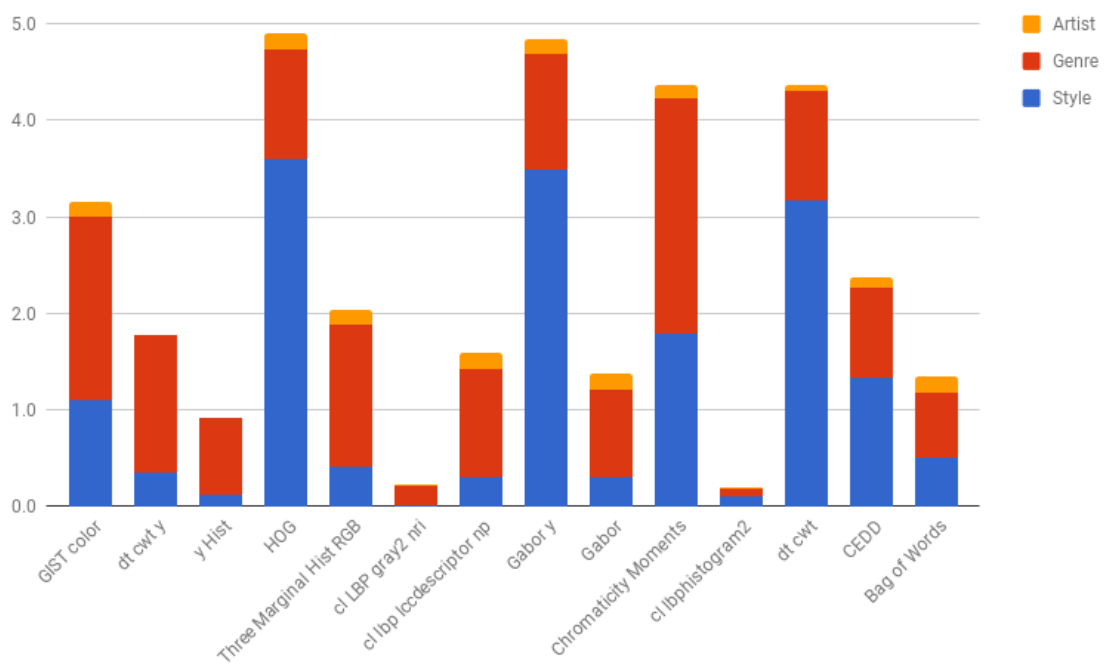


Fig. 6.9 Percentage of correctly classified examples out of all misclassified from our base neural network. Stacked bar chart for the three tasks together. HOG and Gabor y give the highest improvement.

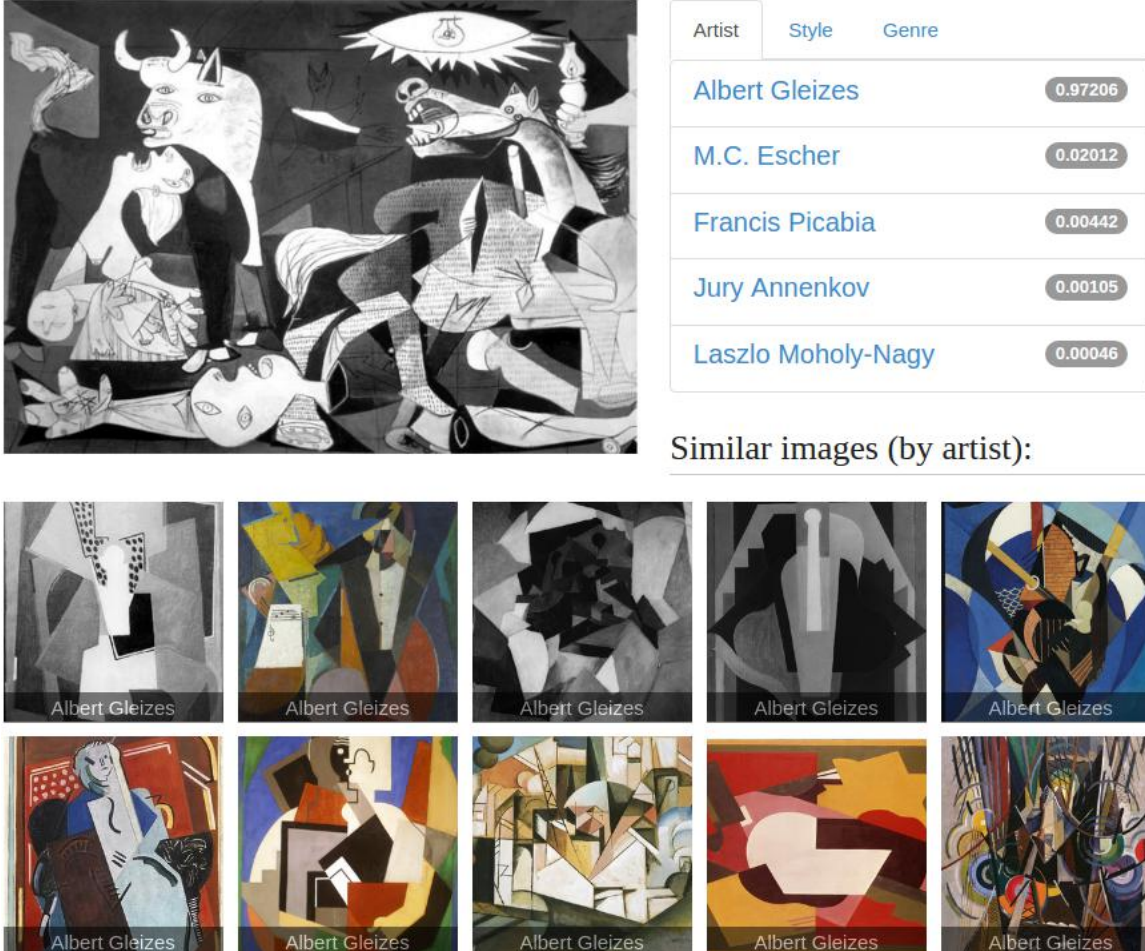
to search in the training set the sample that mostly activate a particular neuron. A similar approach has been adopted to better understand decisions and errors of the proposed architecture based on *Similarity Search*.

The activation vector from the last fully-connected layer is extracted and compared using an l_2 distance to the vectors extracted from all the training set images. Paintings are ordered by distance and the nearest are shown. The proposed architecture is a multitask network trained to predict artist, style and genre of the paintings, thus there are three different final fully-connected layers near the output. Each layer codifies different information about the three separate tasks. Hence it is possible to rank the training set images by three different types of similarity: by artist, by style and by genre.

Figure 6.10 shows an example interface of the similarity search tool. Top-left is the input painting. Top right are shown the five classes with highest probabilities. In the interface there are three tabs, one for each task. In Figure 6.10 only predictions for artist are shown. The bottom part shows the most similar paintings (by artist) retrieved from the training set.

In this particular case the network classifies the painting “Guernica” by Pablo Picasso incorrectly attributing it to the painter Albert Gleizes. The interesting thing to notice is that, among paintings in the first row, three of them are made by composition of squared shapes and present similar colors to Guernica. However, the network predicted the correct style. Gleizes was one of the leading members of cubism and was coeval of Picasso. Probably they also met and influenced each others.

Painting Categorization



Artist	Style	Genre
Albert Gleizes		0.97206
M.C. Escher		0.02012
Francis Picabia		0.00442
Jury Annenkov		0.00105
Laszlo Moholy-Nagy		0.00046

Similar images (by artist):




Fig. 6.10 Screen-shot of the proposed similarity search system used as a visual debug tool to understand network prediction errors. In this case a Picasso painting is incorrectly attributed to Gleizes but images retrieved from the training set can help understand system behavior.

Chapter 7

Conclusions

In this thesis an in-depth study of detectors and descriptors for object recognition has been presented. Two main categories of algorithms have been discussed: Keypoint-based approaches and Convolutional Neural Networks. In the first part several state-of-the-art Keypoint-based methods have been evaluated for the task of Visual Search. Grayscale and color descriptors have been compared on a set of standard MPEG datasets and on a new use case where color is a fundamental clue to discriminate objects. In the second part Keypoint-based approaches have been compared to Convolutional Neural Networks to tackle the problem of Logo recognition. CNNs showed better performances especially when dealing with high intensity image distortions. Finally the problem of Painting Categorization has been investigated. A novel Neural Network architecture has been specifically tailored to jointly use neural features and handcrafted approaches.

In Chapter 3 a detailed analysis of thirteen gray-level interest point detectors and descriptors available in the state of art has been performed on six heterogeneous datasets using a pairwise matching procedure adopted by the MPEG CDVS Test Model [5]. Lowe’s SIFT [139] was confirmed, without a-priori knowledge of the dataset, as the best performing method on average in terms of TPR levels (average TPR: 0.91) among gray-level descriptors. Remarkably, KAZE and SURF performed well on some datasets. Also the affine-invariant detectors achieved good results on the “Landmarks and Buildings” dataset but their results decreased if the number of keypoints detected is limited.

The measured computational times showed that the algorithm with the best trade-off between performance and speed was SIFT, in particular the OpenCV implementation. This was the base complexity adopted by MPEG CDVS against which any color descriptor algorithm has to be measured in terms of extra computational burden added. The use of color information did not achieve interesting performances on all CDVS

Conclusions

datasets. This was proven by the fact that the average TPR value of RGB SIFT which was the best performing algorithm showed on average a 1% improvement with respect to the gray-level baseline. This gain was too limited to justify the extra complexity added by the Early Fusion methods. Color descriptors proven to bring some improvements on datasets 5 (“Common objects and scenes”) and 6 (“SuperMarket Milan”) where color information was certainly more relevant. On dataset 5, Late Fusion algorithms achieved the best TPR values while on dataset 6 the best performing algorithms were some Early Fusion approaches: RGB, Transformed Color, and Opponent. The concern for extra complexity mainly due to look-up handling justified their usage on top of the SIFT complexity baseline.

In Chapter 5 the problem of logo recognition has been investigated. This is usually addressed with keypoint-based methods on high-quality images. A Convolutional Neural Networks is used instead as a robust alternative for low-quality images. The proposed pipeline involved selecting candidate sub-windows using Selective Search, augmenting the training set using Transformation Pursuit, and performing Query Expansion for increasing recall. The method proved to be effective even with CNN features that were trained for a different task, producing results close to the state of the art keypoint-based approaches. The robustness of the method has been investigated with respect to three different kinds of distortion: blur, noise and lossy compression. Results showed that noise was the most affecting one, while lossy compression produced little to no performance loss.

In the second part of Chapter 5 an improved processing pipeline has been proposed. The solution employs a CNN specifically trained for the task of logo classification, even if they are not perfectly localized. A complete recognition pipeline including a recall-oriented candidate logo region proposal has been designed.

Experiments have been carried out on the FlickrLogos-32 database and on its enlarged version, Logos-32plus. The effect on recognition performance of synthetic versus real data augmentation is systematically investigated together with image pre-processing, and the benefits of different training choices such as class-balancing, sample-weighting and explicit modeling the background class (i.e. no-logo regions). The best proposed solution outperforms methods in the state of the art and makes use of an explicit modeling of the background class, both precise and actual object-proposal logo annotations during training, synthetic data augmentation, epoch-based class balancing, and image contrast normalization as pre-processing, while sample weighting is disabled.

In Chapter 6 a novel approach to accomplish the task of painter and style recognition has been proposed. Two different crop strategies are adopted allowing to exploit multiple

cues at different scales. The first permits to manage the trade off between accuracy and speed adopting a random crop strategy and the second is a deterministic extractor based on Spatial Transformer Networks. The crops are fed to a multibranch deep neural network which merge the information at multiple scales and different spatial locations and performs the final prediction. A comparison with state-of-the art methods has been carried out on Paintings-91 dataset for the tasks of Artist and Style recognition. The proposed CNN architecture clearly outperforms previous methods by a large margin.

A new dataset named Wikipaintings-IVL dataset has been collected to perform multitask classification. The new dataset contains about 100K images compared to 4K of Paintings-91. On this dataset different types of multibranch architectures have been evaluated like the multitask networks and the smart crop extractor based on Spatial Transform Networks and also an hybrid solution including multitask CNN and handcrafted features injection. A set of experiments have been made to quantify the discriminative power of handcrafted features on Wikipaintings-IVL dataset. The most promising descriptor is HOG. In a further experiment HOG features has been injected into the Neural Network architecture giving a performances improvement to the overall pipeline.

Finally a novel data augmentation method is introduced. By exploiting neural style transfer technique discussed in Appendix A new fake paintings are produced. Results of two experiments showed that the proposed data augmentation technique acts as a form of regularization.

An interesting topic for future investigations concerns the joint use of Convolutional Neural Networks and handcrafted features [18]. In this thesis, in particular for logo recognition, Neural Networks have been used as an heavy classification model targeted to high accuracy. Images have been previously processed by a Selective Search algorithm that selects the most promising candidate regions. Future works could investigate improved types of synergy between the two categories of descriptors. Furthermore the approaches proposed in this thesis focused on object recognition in still images. Directions for future works could be the application to videos where constraints on consistency between frames could be exploited to enhance the system. In this perspective, a joint use of handcrafted features and Convolutional Neural Networks could benefit for example from the use of optical flow algorithms.

References

- [res] Resnet experiments on torch blog. <http://torch.ch/blog/2016/02/04/resnets.html>. Accessed: 2017-11-14.
- [UDC] Udacity deep learning course. https://storage.googleapis.com/supplemental_media/udacityu/5452470513/Gradient%20Descent.pdf. Accessed: 2017-11-26.
- [UFL] Ufdl deep learning stanford tutorial. <http://ufdl.stanford.edu/tutorial/supervised/ConvolutionalNeuralNetwork/>. Accessed: 2017-11-14.
- [4] (2011). Evaluation framework for compact descriptors for visual search. ISO/IEC JTC1/SC29/WG11/N12202, July 2011, Torino, IT.
- [5] (2014). MPEG 108 CDVS Test Model 10: Compact descriptors for visual search. ISO/IEC JTC1/SC29/WG11 MPEG2014 W14393, April 2014, Valencia, Spain.
- [6] Abdel-Hakim, A. E. and Farag, A. A. (2006). Csift: A sift descriptor with color invariant characteristics. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 1978–1983. IEEE.
- [7] Alahi, A., Ortiz, R., and Vandergheynst, P. (2012a). Freak: Fast retina keypoint. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 510–517. IEEE.
- [8] Alahi, A., Ortiz, R., and Vandergheynst, P. (2012b). Freak: Fast retina keypoint. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 510–517. Ieee.
- [9] Alcantarilla, P. F., Bartoli, A., and Davison, A. J. (2012a). Kaze features. In *Proceedings of the 12th European conference on Computer Vision - Volume Part VI, ECCV'12*, pages 214–227, Berlin, Heidelberg. Springer-Verlag.
- [10] Alcantarilla, P. F., Bartoli, A., and Davison, A. J. (2012b). Kaze features. In *Computer Vision–ECCV 2012*, pages 214–227. Springer.
- [11] Alcantarilla, P. F., Nuevo, J., and Bartoli, A. (2013). Fast explicit diffusion for accelerated features in nonlinear scale spaces. In *Proceedings of the British Machine Vision Conference*. BMVA Press.
- [12] Anwer, R. M., Khan, F. S., van de Weijer, J., and Laaksonen, J. (2016). Combining holistic and part-based deep representations for computational painting categorization. In *Proceedings of the 2016 ACM on International Conference on Multimedia Retrieval*, pages 339–342. ACM.

References

- [13] Bagdanov, A. D., Ballan, L., Bertini, M., and Del Bimbo, A. (2007). Trademark matching and retrieval in sports video databases. In *Proceedings of the international workshop on Workshop on multimedia information retrieval*, pages 79–86. ACM.
- [14] Barilla, M. and Spann, M. (2008). Colour-based texture image classification using the complex wavelet transform. In *Electrical Engineering, Computing Science and Automatic Control, 2008. CCE 2008. 5th International Conference on*, pages 358–363.
- [15] Bay, H., Tuytelaars, T., and Gool, L. (2006a). Surf: Speeded up robust features. In *Computer Vision, ECCV 2006*, volume 3951 of *Lecture Notes in Computer Science*, pages 404–417. Springer Berlin Heidelberg.
- [16] Bay, H., Tuytelaars, T., and Van Gool, L. (2006b). Surf: Speeded up robust features. In *Computer Vision—ECCV 2006*, pages 404–417. Springer.
- [17] Benavente, R., Vanrell, M., and Baldrich, R. (2008). Parametric fuzzy sets for automatic color naming. *JOSA A*, 25(10):2582–2593.
- [18] Bianco, S. (2017). Large age-gap face verification by feature injection in deep networks. *Pattern Recognition Letters*, 90:36–42.
- [19] Bianco, S., Buzzelli, M., Mazzini, D., and Schettini, R. (2015). Logo recognition using cnn features. In *Image Analysis and Processing—ICIAP 2015*, pages 438–448. Springer.
- [20] Bianco, S., Buzzelli, M., Mazzini, D., and Schettini, R. (2017). Deep learning for logo recognition. *Neurocomputing*.
- [21] Bianco, S. and Cusano, C. (2011). Color target localization under varying illumination conditions. In *Computational Color Imaging*, pages 245–255. Springer.
- [22] Bianco, S., Schettini, R., Mazzini, D., and Pau, D. P. (2015 (accepted)). Local detectors and compact descriptors: a quantitative comparison. *Digital Signal Processing*.
- [23] Bianconi, F. and Fernández, A. (2007). Evaluation of the effects of gabor filter parameters on texture classification. *Pattern Recognition*, 40(12):3325 – 3335.
- [24] Bianconi, F., Harvey, R., Southam, P., and Fernández, A. (2011). Theoretical and experimental comparison of different approaches for color texture classification. *Journal of Electronic Imaging*, 20(4).
- [25] Boia, R., Florea, C., and Florea, L. (2015). Elliptical asift agglomeration in class prototype for logo detection. In *BMVC*, pages 115–1.
- [26] Boia, R., Florea, C., Florea, L., and Dogaru, R. (2016). Logo localization and recognition in natural images using homographic class graphs. *Machine Vision and Applications*, 27(2):287–301.
- [27] Bradski, G. (2000). The opencv library. *Doctor Dobbs Journal*, 25(11):120–126.

-
- [28] Bratkova, M., Boulos, S., and Shirley, P. (2009). oRGB: a practical opponent color space for computer graphics. *IEEE Computer Graphics and Applications*, 29(1):42–55.
- [29] Canziani, A., Paszke, A., and Culurciello, E. (2016). An analysis of deep neural network models for practical applications. *arXiv preprint arXiv:1605.07678*.
- [30] Carneiro, G., da Silva, N. P., Del Bue, A., and Costeira, J. P. (2012). Artistic image classification: An analysis on the printart database. In *European Conference on Computer Vision*, pages 143–157. Springer.
- [31] Chambolle, A. (2004). An algorithm for total variation minimization and applications. *Journal of Mathematical imaging and vision*, 20(1):89–97.
- [32] Cimpoi, M., Maji, S., and Vedaldi, A. (2015). Deep filter banks for texture recognition and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3828–3836.
- [33] Cireřan, D. C., Meier, U., Masci, J., Gambardella, L. M., and Schmidhuber, J. (2011). High-performance neural networks for visual object classification. *arXiv preprint arXiv:1102.0183*.
- [34] Cortes, C. and Vapnik, V. (1995). Support vector machine. *Machine learning*, 20(3):273–297.
- [35] Cusano, C., Napoletano, P., and Schettini, R. (2014). Combining local binary patterns and local color contrast for texture classification under varying illumination. *JOSA A*, 31(7):1453–1461.
- [36] Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems (MCSS)*, 2(4):303–314.
- [37] d’Angelo, E., Alahi, A., and Vandergheynst, P. (2012). Beyond bits: Reconstructing images from local binary descriptors. In *Pattern Recognition (ICPR), 2012 21st International Conference on*, pages 935–938. IEEE.
- [38] Dhillon, I. S., Mallela, S., and Kumar, R. (2003). A divisive information theoretic feature clustering algorithm for text classification. *The Journal of Machine Learning Research*, 3:1265–1287.
- [39] Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159.
- [40] Eggert, C., Winschel, A., and Lienhart, R. (2015). On the benefit of synthetic data for company logo detection. In *Proceedings of the 23rd Annual ACM Conference on Multimedia Conference*, pages 1283–1286. ACM.
- [41] Felzenszwalb, P. F. and Huttenlocher, D. P. (2004). Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181.

References

- [42] Fierro, R. and Lewis, F. (1999). Multilayer feedforward networks are universal approximators. *IEEE Trans. Syst., Man, Cybern.*, 29(6):649–654.
- [43] Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 36(4):193–202.
- [44] Gao, Y., Wang, F., Luan, H., and Chua, T.-S. (2014). Brand data gathering from live social media streams. In *Proceedings of International Conference on Multimedia Retrieval*, page 169. ACM.
- [45] Gatys, L., Ecker, A. S., and Bethge, M. (2015a). Texture synthesis using convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 262–270.
- [46] Gatys, L. A., Ecker, A. S., and Bethge, M. (2015b). A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576*.
- [47] Gatys, L. A., Ecker, A. S., and Bethge, M. (2016). Image style transfer using convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2414–2423.
- [48] Geusebroek, J. M., Burghouts, G. J., and Smeulders, A. W. M. (2005). The amsterdam library of object images. *International Journal of Computer Vision*, 61(1):103–112.
- [49] Geusebroek, J.-M., Van den Boomgaard, R., Smeulders, A. W. M., and Geerts, H. (2001). Color invariance. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(12):1338–1350.
- [50] Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587.
- [51] Gong, Y., Wang, L., Guo, R., and Lazebnik, S. (2014). Multi-scale orderless pooling of deep convolutional activation features. In *European Conference on Computer Vision*, pages 392–407. Springer.
- [52] Grady, C. L., McIntosh, A. R., Rajah, M. N., and Craik, F. I. (1998). Neural correlates of the episodic encoding of pictures and words. *Proceedings of the National Academy of Sciences*, 95(5):2703–2708.
- [53] Graf, A. B. and Borer, S. (2001). Normalization in support vector machines. In *Pattern Recognition: 23rd DAGM Symposium, Munich, Germany, September 12-14, 2001. Proceedings*, page 277. Springer.
- [54] Hagbi, N., Bergig, O., El-Sana, J., and Billingham, M. (2011). Shape recognition and pose estimation for mobile augmented reality. *Visualization and Computer Graphics, IEEE Transactions on*, 17(10):1369–1379.
- [55] Hariharan, B., Malik, J., and Ramanan, D. (2012). Discriminative decorrelation for clustering and classification. *Computer Vision–ECCV 2012*, pages 459–472.

-
- [56] Harris, C. and Stephens, M. (1988). A Combined Corner and Edge Detection. In *Proceedings of The Fourth Alvey Vision Conference*, pages 147–151.
- [57] He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034.
- [58] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778.
- [59] Hentschel, C., Wiradarma, T. P., and Sack, H. (2016). Fine tuning cnns with scarce training data—adapting imagenet to art epoch classification. In *Image Processing (ICIP), 2016 IEEE International Conference on*, pages 3693–3697. IEEE.
- [60] Hosang, J., Benenson, R., Dollár, P., and Schiele, B. (2016). What makes for effective detection proposals? *IEEE transactions on pattern analysis and machine intelligence*, 38(4):814–830.
- [61] Iandola, F. N., Shen, A., Gao, P., and Keutzer, K. (2015). Deeplogo: Hitting logo recognition with the deep neural network hammer. *arXiv preprint arXiv:1510.02131*.
- [62] Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of The 32nd International Conference on Machine Learning*, pages 448–456.
- [63] Isola, P., Xiao, J., Torralba, A., and Oliva, A. (2011). What makes an image memorable? In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 145–152. IEEE.
- [64] Jaderberg, M., Simonyan, K., Zisserman, A., et al. (2015). Spatial transformer networks. In *Advances in Neural Information Processing Systems*, pages 2017–2025.
- [65] Jégou, H., Douze, M., Schmid, C., and Pérez, P. (2010). Aggregating local descriptors into a compact image representation. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 3304–3311.
- [66] Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., and Darrell, T. (2014). Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 675–678. ACM.
- [67] Johnson, J., Alahi, A., and Fei-Fei, L. (2016). Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision*, pages 694–711. Springer.
- [68] Joly, A. and Buisson, O. (2009). Logo retrieval with a contrario visual query expansion. In *Proceedings of the 17th ACM international conference on Multimedia*, pages 581–584. ACM.

References

- [69] Junior, O. L., Delgado, D., Gonçalves, V., and Nunes, U. (2009). Trainable classifier-fusion schemes: an application to pedestrian detection. In *Intelligent Transportation Systems*.
- [70] Khan, F. S., Beigpour, S., Van de Weijer, J., and Felsberg, M. (2014). Painting-91: a large scale database for computational painting categorization. *Machine vision and applications*, 25(6):1385–1397.
- [71] Khan, F. S., van de Weijer, J., and Vanrell, M. (2012). Modulating shape features by color attention for object recognition. *International Journal of Computer Vision*, 98(1):49–64.
- [72] Khan, R., Van de Weijer, J., Khan, F. S., Muselet, D., Ducottet, C., and Barat, C. (2013). Discriminative color descriptors. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 2866–2873. IEEE.
- [73] Kingma, D. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.
- [74] Kleban, J., Xie, X., and Ma, W.-Y. (2008). Spatial pyramid mining for logo detection in natural scenes. In *Multimedia and Expo, 2008 IEEE International Conference on*, pages 1077–1080. IEEE.
- [75] Koenderink, J. (1984). The structure of images. *Biological cybernetics*, 50(5):363–370.
- [76] 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.
- [77] 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.
- [78] Ledig, C., Theis, L., Huszár, F., Caballero, J., Cunningham, A., Acosta, A., Aitken, A., Tejani, A., Totz, J., Wang, Z., et al. (2016). Photo-realistic single image super-resolution using a generative adversarial network. *arXiv preprint*.
- [79] Leutenegger, S., Chli, M., and Siegwart, R. Y. (2011). Brisk: Binary robust invariant scalable keypoints. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2548–2555. IEEE.
- [80] Lindeberg, T. (1994). Scale-space theory: A basic tool for analysing structures at different scales. *Journal of Applied Statistics*, pages 224–270.
- [81] Lindeberg, T. (1998). Feature detection with automatic scale selection. *International Journal of Computer Vision*, 30:79–116.
- [82] Lowe, D. (1999a). Object recognition from local scale-invariant features. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 2, pages 1150–1157 vol.2.

-
- [83] Lowe, D. (2004a). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110.
- [84] Lowe, D. G. (1999b). Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee.
- [85] Lowe, D. G. (2004b). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110.
- [86] Mäenpää, T. and Pietikäinen, M. (2004). Classification with color and texture: jointly or separately? *Pattern Recognition*, 37(8):1629–1640.
- [87] Mahendran, A. and Vedaldi, A. (2015). Understanding deep image representations by inverting them. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5188–5196.
- [88] Mair, E., Hager, G., Burschka, D., Suppa, M., and Hirzinger, G. (2010). Adaptive and generic corner detection based on the accelerated segment test. *Computer Vision–ECCV 2010*, pages 183–196.
- [89] McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133.
- [90] Meng, J., Yuan, J., Jiang, Y., Narasimhan, N., Vasudevan, V., and Wu, Y. (2010). Interactive visual object search through mutual information maximization. In *Proceedings of the 18th ACM international conference on Multimedia*, pages 1147–1150. ACM.
- [91] Mensink, T. and Van Gemert, J. (2014). The rijksmuseum challenge: Museum-centered visual recognition. In *Proceedings of International Conference on Multimedia Retrieval*, page 451. ACM.
- [92] Mikolajczyk, K. and Schmid, C. (2002). An affine invariant interest point detector. In *Proceedings of the 7th European Conference on Computer Vision-Part I, ECCV '02*, pages 128–142, London, UK, UK. Springer-Verlag.
- [93] Mikolajczyk, K. and Schmid, C. (2004a). Scale & affine invariant interest point detectors. *International Journal of Computer Vision*, 60(1):63–86.
- [94] Mikolajczyk, K. and Schmid, C. (2004b). Scale and affine invariant interest point detectors. *International Journal of Computer Vision*, 60(1):63–86.
- [95] Mikolajczyk, K. and Schmid, C. (2005). A performance evaluation of local descriptors. page 34.
- [96] Mikolajczyk, K., Tuytelaars, T., Schmid, C., Zisserman, A., Matas, J., Schaffalitzky, F., Kadir, T., and Gool, L. V. (2005). A comparison of affine region detectors. *International Journal of Computer Vision*, 65:2005.
- [97] Moravec, H. P. (1977). Towards automatic visual obstacle avoidance. In *IJCAI*, page 584.

References

- [98] Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814.
- [99] Nam, W., Dollár, P., and Han, J. H. (2014). Local decorrelation for improved pedestrian detection. In *Advances in Neural Information Processing Systems*, pages 424–432.
- [100] Novak, C. L., Shafer, S., et al. (1992). Anatomy of a color histogram. In *Computer Vision and Pattern Recognition, 1992. Proceedings CVPR'92., 1992 IEEE Computer Society Conference on*, pages 599–605. IEEE.
- [101] Oliva, A. and Torralba, A. (2001). Modeling the shape of the scene: A holistic representation of the spatial envelope. *Int'l J. Computer Vision*, 42(3):145–175.
- [102] Oliveira, G., Frazão, X., Pimentel, A., and Ribeiro, B. (2016). Automatic graphic logo detection via fast region-based convolutional networks. In *Neural Networks (IJCNN), 2016 International Joint Conference on*, pages 985–991. IEEE.
- [103] Parks, D. and Gravel, J.-P. (2004). Corner detection. *International Journal of Computer Vision*, pages 1–17.
- [104] Paschos, G. (2000). Fast color texture recognition using chromaticity moments. *Pattern Recognition Letters*, 21(9):837 – 841.
- [105] Pau, D., Cordara, G., Bober, M., Paschalakis, S., Iwamoto, K., Francini, G., Chandrasekhar, V., and Takacs, G. (2013). White paper on compact descriptors for visual search. ISO/IEC JTC1/SC29/WG11 MPEG2013/N13951, Incheon, Korea, April 2013.
- [106] Paulin, M., Revaud, J., Harchaoui, Z., Perronnin, F., and Schmid, C. (2014). Transformation pursuit for image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3646–3653.
- [107] Peng, K.-C. and Chen, T. (2015a). Cross-layer features in convolutional neural networks for generic classification tasks. In *Image Processing (ICIP), 2015 IEEE International Conference on*, pages 3057–3061. IEEE.
- [108] Peng, K.-C. and Chen, T. (2015b). A framework of extracting multi-scale features using multiple convolutional neural networks. In *2015 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6. IEEE.
- [109] Perona, P. and Malik, J. (1990). Scale-space and edge detection using anisotropic diffusion. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 12(7):629–639.
- [110] Pietikainen, M., Nieminen, S., Marszalec, E., and Ojala, T. (1996). Accurate color discrimination with classification based on feature distributions. In *Pattern Recognition, 1996., Proceedings of the 13th International Conference on*, volume 3, pages 833 –838 vol.3.

-
- [111] Psyllos, A. P., Anagnostopoulos, C.-N. E., and Kayafas, E. (2010). Vehicle logo recognition using a sift-based enhanced matching scheme. *Intelligent Transportation Systems, IEEE Transactions on*, 11(2):322–328.
- [112] Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99.
- [113] Revaud, J., Douze, M., and Schmid, C. (2012). Correlation-based burstiness for logo retrieval. In *Proceedings of the 20th ACM international conference on Multimedia*, pages 965–968. ACM.
- [114] Romberg, S. and Lienhart, R. (2013). Bundle min-hashing for logo recognition. In *Proceedings of the 3rd ACM conference on International conference on multimedia retrieval*, pages 113–120. ACM.
- [115] Romberg, S., Pueyo, L. G., Lienhart, R., and Van Zwol, R. (2011). Scalable logo recognition in real-world images. In *Proceedings of the 1st ACM International Conference on Multimedia Retrieval*, page 25. ACM.
- [116] Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386.
- [117] Rosenfeld, A. (2014). *Human and machine vision II*. Academic Press.
- [118] Rosten, E. and Drummond, T. (2006). Machine learning for high-speed corner detection. In *In European Conference on Computer Vision*, pages 430–443.
- [119] Ruder, M., Dosovitskiy, A., and Brox, T. (2016). Artistic style transfer for videos. In *German Conference on Pattern Recognition*, pages 26–36. Springer.
- [120] Ruder, S. (2017). An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*.
- [121] 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.
- [122] Shahbaz Khan, F., Anwer, R. M., van de Weijer, J., Bagdanov, A. D., Vanrell, M., and Lopez, A. M. (2012). Color attributes for object detection. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3306–3313. IEEE.
- [123] Sharif 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.
- [124] Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

References

- [Sinha] Sinha, U. Ai shack: <http://www.aishack.in>.
- [126] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.
- [127] Srivastava, R. K., Greff, K., and Schmidhuber, J. (2015a). Highway networks. *arXiv preprint arXiv:1505.00387*.
- [128] Srivastava, R. K., Greff, K., and Schmidhuber, J. (2015b). Training very deep networks. In *Advances in neural information processing systems*, pages 2377–2385.
- [129] 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.
- [130] Tan, W. R., Chan, C. S., Aguirre, H. E., and Tanaka, K. (2016). Ceci n’est pas une pipe: A deep convolutional network for fine-art paintings classification. In *Image Processing (ICIP), 2016 IEEE International Conference on*, pages 3703–3707. IEEE.
- [131] Tsai, S. S., Chen, D., Takacs, G., Chandrasekhar, V., Vedantham, R., Grzeszczuk, R., and Girod, B. (2010). Fast geometric re-ranking for image-based retrieval. In *Image Processing (ICIP), 2010 17th IEEE International Conference on*, pages 1029–1032. IEEE.
- [132] Uijlings, J. R. R., van de Sande, K. E. A., Gevers, T., and Smeulders, A. W. M. (2013). Selective search for object recognition. *International Journal of Computer Vision*, 104(2):154–171.
- [133] Ulyanov, D., Lebedev, V., Vedaldi, A., and Lempitsky, V. S. (2016). Texture networks: Feed-forward synthesis of textures and stylized images. In *ICML*, pages 1349–1357.
- [134] Van De Sande, K. E., Gevers, T., and Snoek, C. G. (2010). Evaluating color descriptors for object and scene recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(9):1582–1596.
- [135] van de Sande, K. E. A., Gevers, T., and Snoek, C. G. M. (2009). Evaluating color descriptors for object and scene recognition. *Transactions on Pattern Analysis and Machine Intelligence*.
- [136] van de Sande, K. E. A., Uijlings, J. R. R., Gevers, T., and Smeulders, A. W. M. (2011). Segmentation as selective search for object recognition. In *IEEE International Conference on Computer Vision*.
- [137] Van De Weijer, J. and Schmid, C. (2007). Applying color names to image description. In *Image Processing, 2007. ICIP 2007. IEEE International Conference on*, volume 3, pages III–493. IEEE.

-
- [138] Van De Weijer, J., Schmid, C., Verbeek, J., and Larlus, D. (2009). Learning color names for real-world applications. *Image Processing, IEEE Transactions on*, 18(7):1512–1523.
- [139] Vedaldi, A. and Fulkerson, B. (2010). Vlfeat: An open and portable library of computer vision algorithms. In *Proceedings of the International Conference on Multimedia*, pages 1469–1472. ACM.
- [140] Verma, A., Banerji, S., and Liu, C. (2010). A new color sift descriptor and methods for image category classification. In *International Congress on Computer Applications and Computational Science*, pages 4–6.
- [141] Viola, P. and Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–511 – I–518 vol.1.
- [142] von Kries, J. (1970). Influence of adaptation on the effects produced by luminous stimuli.
- [143] Wan, L., Zeiler, M., Zhang, S., Cun, Y. L., and Fergus, R. (2013). Regularization of neural networks using dropconnect. In *Proceedings of the 30th international conference on machine learning (ICML-13)*, pages 1058–1066.
- [144] Wang, X., Hou, Z., Yu, W., Xue, Y., Jin, Z., and Dai, B. (2017). Robust visual tracking via multiscale deep sparse networks. *Optical Engineering*, 56(4):043107–043107.
- [145] Wang, Z., Bovik, A. C., Sheikh, H. R., and Simoncelli, E. P. (2004). Image quality assessment: from error visibility to structural similarity. *Image Processing, IEEE Transactions on*, 13(4):600–612.
- [146] Weickert, J. (2001). Efficient image segmentation using partial differential equations and morphology. *Pattern Recognition*, 34(9):1813–1824.
- [147] Widjaja, I., Leow, W. K., and Wu, F.-C. (2003). Identifying painters from color profiles of skin patches in painting images. In *Image Processing, 2003. ICIP 2003. Proceedings. 2003 International Conference on*, volume 1, pages I–845. IEEE.
- [148] Zeiler, M. D. A. (2012). An adaptive learning rate method. arxiv preprint. *arXiv preprint arXiv:1212.5701*.
- [149] Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. (2016). Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*.
- [150] Zhu, C., Byrd, R. H., Lu, P., and Nocedal, J. (1994). Lbfgs-b: Fortran subroutines for large-scale bound constrained optimization. *Report NAM-11, EECS Department, Northwestern University*.

Appendix A

Texture Synthesis and Style Transfer

Painting dataset introduced in Section 6.3.3 has been augmented by generating synthetic paintings. The process of generating paintings has been carried out with a technique based on Neural Style Transfer [46], [47]. This algorithm make use of a pretrained network to combine texture and coarse shapes from two arbitrary images. In this Appendix first is introduced the texture synthesis algorithm first presented in [45], then is presented the neural style transfer technique in two variants: optimization-based (slow), feed-forward (fast).

A.1 Texture synthesis

Goal of texture synthesis is to build a generative model able to capture the characteristic distribution of an input texture to produce new texture perceptually similar or ideally

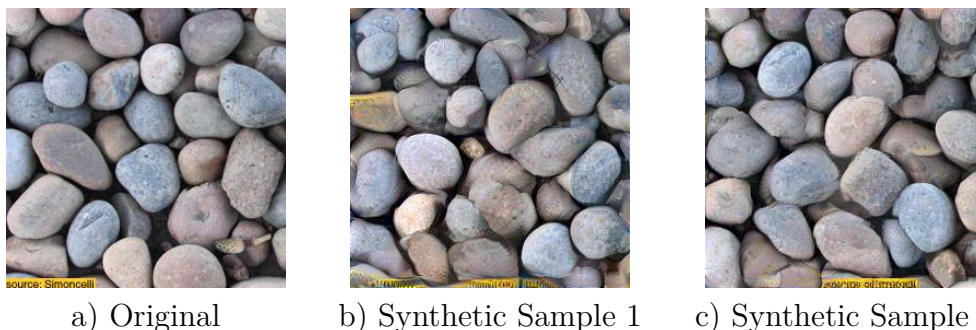


Fig. A.1 Examples of synthetic textures produced by the model presented in Section A.1. Both synthetic textures are generated solely from texture a.

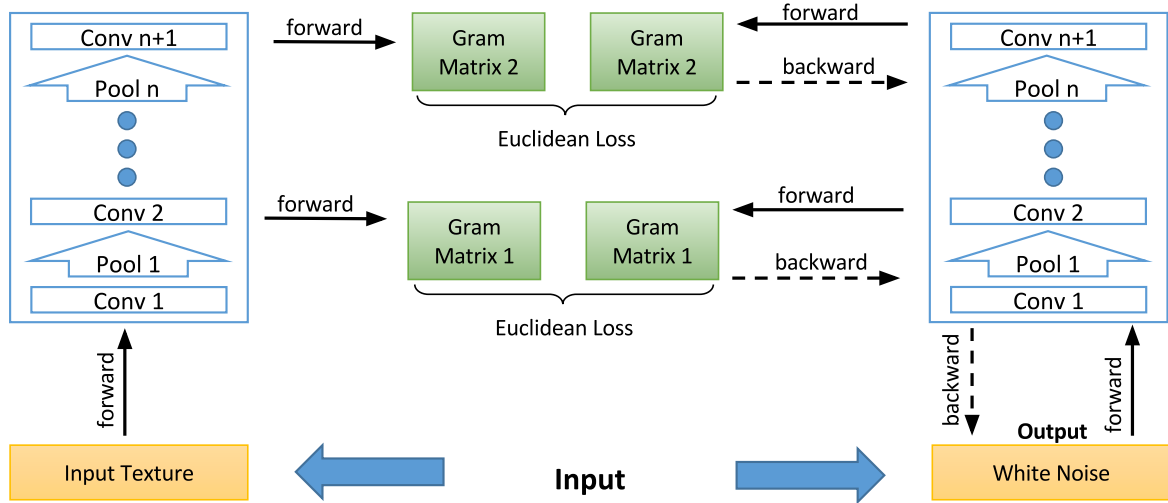


Fig. A.2 Texture synthesis model of Gatys et al. [45]. It is composed by two siblings deep networks, the first fed with the texture image and the second with white noise. The white noise is shaped to match the same gram matrices statistics of the original texture.

undistinguishable from the source texture. In 2015 Gatys et al. presented the first work on texture synthesis using Convolutional Neural Networks [45]. The idea is simple and effective: using a powerful pretrained Neural Network not only as feature extractor but to induce a particular distribution of pixels in the 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 layers near the random input image.

Figure A.1 shows two samples of synthetic textures generated with Gatys model. Features are extracted using a VGG network (see Section 4.6) from activations (also called feature-maps) at different layers of the network. Correlations are computed on the extracted features by means of Gram matrices. A Gram matrix is computed by:

$$G_{ij}^l = \sum_k F_{ik}^l F_{jk}^l \quad (\text{A.1})$$

where $F^l \in \mathbb{R}^{N_l \times M_l}$ is a vector matrix obtained from a set of feature-maps of size N_l by flattening the spatial dimension to a single dimension M_l . F_{jk}^l represent the activation of the j^{th} filter at position k in layer l . The Gram matrix contains only information about correlations between feature-maps discarding spatial information,

forcing a stationary description of the input image. This description fully specifies a texture in the model proposed by Gatys et al. [45].

Texture synthesis is carried out by the backpropagation algorithm through a clone of the VGG network used for feature extraction. The network is fed with a white noise image. Gram matrices are computed from feature-maps at various levels. Finally these gram matrices are compared with gram matrices produced by the original texture by means of an Euclidean loss and the resulting gradients are backpropagated through the network to shape the white noise image. Figure A.2 shows the whole synthesis system. Given \vec{x} and $\hat{\vec{x}}$ as the original image and the white noise, a single loss E_l obtained by comparing two gram matrices has the following form:

$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} (G_{ij}^l - \hat{G}_{ij}^l)^2 \quad (\text{A.2})$$

where G and \hat{G} are the two Gram matrices computed from the texture and the white noise image respectively. The total loss is given by:

$$\mathcal{L}_{texture}(\vec{x}, \hat{\vec{x}}) = \sum_{l=0}^L w_l E_l \quad (\text{A.3})$$

where l is the layer index. The overall loss is a weighted sum of Euclidean losses over each Gram matrix pair. Figure A.3 shows a synthesis of the rocks texture in Figure

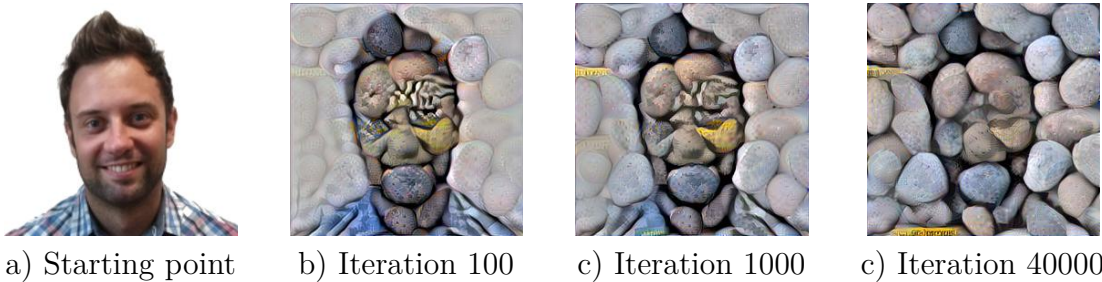


Fig. A.3 Synthesis of the rocks texture in Figure A.1a. The starting point is set to the picture of a face. Intermediate output at different iterations of the algorithm is shown.

A.1a where the starting point is not random noise but a picture of a face. The Figure contains three outputs corresponding to three different iterations of the optimization process. The final appearance of the image is heavily influenced by the initialization. The algorithm tends to place rocks' borders in correspondence of strong edges in the initialization image. The optimization algorithm tends to favor areas with edges, flat areas are the last to be rendered. Another interesting thing to notice is about the yellow

label in the source image in Figure A.1a bottom left. In the final synthetic textures is always rendered near the bottom border. Since spatial information is completely lost when computing Gram matrices, this suggest that, in the VGG network architecture, some neurons are devoted to handle information lying on image borders. Moreover this type of neurons are specialized for each of the four borders.

A.2 Image Style Transfer

The same authors of [45], carried on the work on texture synthesis expanding it to a new application: Image style transfer [47]. The problem of style transfer is reformulated as a problem of texture transfer. They used the same optimization algorithm in Section A.1 with two constraints: the resulting image must have a texture with the same statistics as the source texture and at the same time the same coarse structure as another image named *content image*.

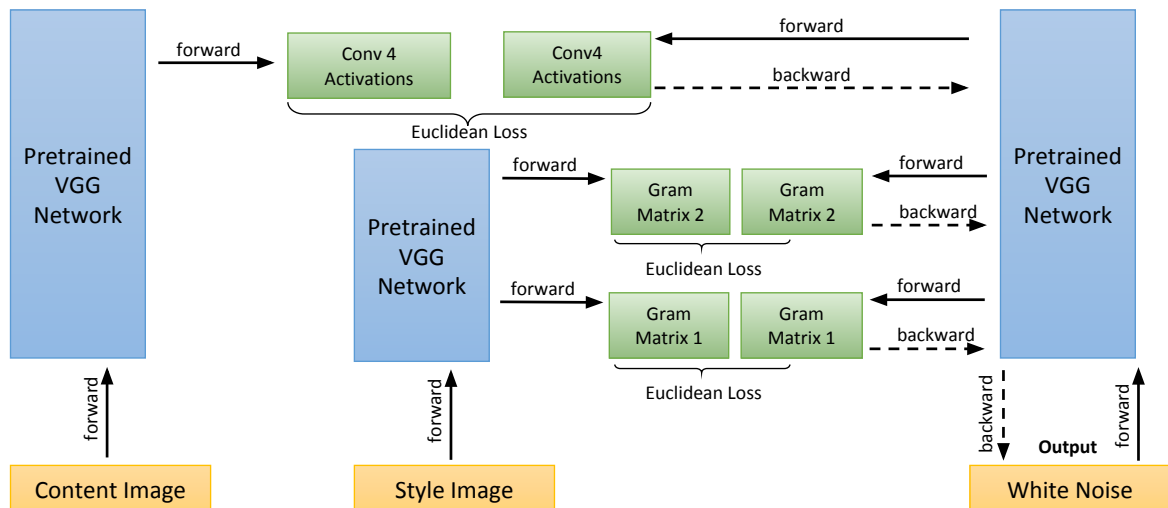


Fig. A.4 Network architecture used to perform style transfer. Two siblings VGG networks are used as feature extractors for content and style. Another network is used to shape the output image initialized as white noise.

Figure A.4 shows the system used to operate style transfer. It takes as input three images: content, style and white noise image i.e. the output. Most part of the system represented in Figure is the same as the one for texture synthesis presented in Figure A.2. The two rightmost networks are the part of the system borrowed from the texture synthesis network. Gram matrices computed from internal activation layers of both networks are matched by means of the Euclidean distance and the resulting gradients flow back trough one network to the white noise image. The main difference is the

addition of a new sibling network fed with the content image. Inner activations of this network are directly compared to those of the network fed with white noise. No use of Gram matrices is adopted because in this case the goal is to retain spatial information from the content image. Moreover activations used to synthesize content are usually higher layers compared to those considered to synthesize texture. This is because they represent more abstract concepts than lower layers. The goal of style transfer is to produce a new image with low-level details (texture) of one image while retaining coarse level structure (content) of another image. Thus the style transfer network is equipped with two types of loss functions: a *Style Loss* which is the same used for texture synthesis, i.e. Equation A.1 and a *Content Loss* which is presented here:

$$\mathcal{L}_{content}(\vec{x}, \hat{\vec{x}}, l) = \frac{1}{2} \sum_{i,j} (F_{ij}^l - \hat{F}_{ij}^l)^2 \quad (\text{A.4})$$

where \vec{x} and $\hat{\vec{x}}$ are the content and output images respectively, F_{ij}^l and \hat{F}_{ij}^l are activations of the l^{th} layer of the two siblings networks fed with content and output images. This loss is essentially an Euclidean loss over activations of the two networks. The overall loss function to minimize is then the sum of Equations A.1 and A.2:

$$\mathcal{L}_{total}(\vec{c}, \vec{t}, \vec{x}) = \alpha \mathcal{L}_{content}(\vec{c}, \vec{x}) + \beta \mathcal{L}_{texture}(\vec{t}, \vec{x}) \quad (\text{A.5})$$

where $\vec{c}, \vec{t}, \vec{x}$ are the content, texture and output images respectively. α and β are the weighting factors to balance content and texture strength in the output image. The gradient with respect to the pixel values of the output image $\frac{\partial \mathcal{L}_{total}}{\partial \vec{x}}$ can be used as input to perform optimization using Stochastic Gradient Descent or related techniques. Authors of [47] experimentally found L-BFGS [150] to be the best method for image synthesis. Figure A.5 shows an example of style transfer where some parameters are varied. First row shows the Content and Style or Texture image. Second row shows a variation of the style weight parameter, i.e. β in Equation A.2 while the content weight (parameter α) is unchanged to 5×10^1 . As expected, the more style weight is increased and the more texture details are introduced from style image perturbing the content image. With $\beta = 10^3$ the original content is almost undistinguishable. In the third row the activations used to compute the Content Loss are changed. Three feature-maps coming from the output of different relu layers have been used: relu3_2, relu4_2 and relu5_2. The use of relu3_2 and relu4_2 produce similar results whereas when using relu5_2 the content is almost lost in favor of texture. This suggest that

layer relu5_2 is too far from the input image and represents abstract shapes that are not useful for the style transfer task.

Last row shows results of varying the weight of the Total Variation (TV) Loss [31]. TV Loss has been used as a regularizer when generating images with CNNs [87] as well as with other descriptors [37]. It is used to suppress noise by inducing smoothing in the output image. Equation below express TV Loss:

$$\mathcal{R}_{V^\beta}(x) = \sum_{i,j} ((x_{i,j+1})^2 + (x_{i+1,j} - x_{i,j})^2)^{\frac{\beta}{2}} \quad (\text{A.6})$$

where $x_{i,j}$ is a single pixel of the input image and β is a parameter influencing loss strength. All the images shown in this Appendix make use of TV Loss together with the other Losses described in this Section. TV loss value is multiplied by a weight factor and added to the overall loss value. The three images in Figure A.5 last row show a variation of the TV loss weight factor. Note that the more is the influence of TV loss and the more random noise is attenuated. With high weight, i.e. rightmost image, noise is spread on image producing wide colored areas.

A.3 Feed-forward network and Perceptual loss

One downside of this style transfer approach is its computational complexity. Since it is based on an optimization process it is quite computationally heavy and it requires long processing time even for very small images. To give an idea it can take almost 5 minutes to process a 512x512 image with an Nvidia K40 GPU. To overcome this problem Ulyanov et al. proposed an extension of the architecture presented in previous sections [133]. Figure A.6 shows the new network architecture. The upper part in Figure A.6 represents the same style transfer network introduced in the previous Section. The main difference is that a new network is added to the whole architecture. Gradient flow is expanded. Originating from the Content and Style losses it goes through the pretrained VGG Network and flow through the input image, labeled in Figure as “Stylized Image” and finally reach a new network that has to be trained. Now the gradient are not only shaping the input image but through it they are shaping the network weights in order to produce the right image.

As a result a new network is trained to produce the stylized image given an arbitrary content image. Note from Figure that at training time the same content image is fed to the system in two different points: the pretrained VGG and the network to be trained. The whole system architecture is used only in training. During the test phase the

A.3 Feed-forward network and Perceptual loss

new network is detached and used as a standalone module to produce stylized images. The new network can produce stylized images with a single forward pass obtaining similar qualitative results but three order of magnitude faster. A 512x512 image can be generated in a few milliseconds by a modern GPU. Figure A.7 shows some examples of images produced by the feedforward network. The architecture composed by the three siblings VGG networks is actually acting as a very complex loss module. For this reason Johnson et al. in their work [67] called this architecture *Perceptual Loss* and used it as a general purpose module to be used in generative models. They applied the Perceptual Loss module not only to train style transfer networks but also to tackle the task of super-resolution from single image. Since then the Perceptual Loss module is used as a strong loss in numerous tasks involving generative models. As an example Ledig et al. [78] jointly used a Perceptual Loss and a Generative Adversarial training environment to generate super-resolution images. In a similar way, Ruder et al. [119] designed a system to perform style transfer on videos by combining the perceptual loss module together with a loss that ensures consistency among video frames.

In Chapter 6 the Perceptual loss and the whole Image Style Transfer algorithm has been used to generate new paintings to perform data augmentation and induce a form of regularization for the painting categorization model. Quantitative and qualitative results are presented in Chapter 6.

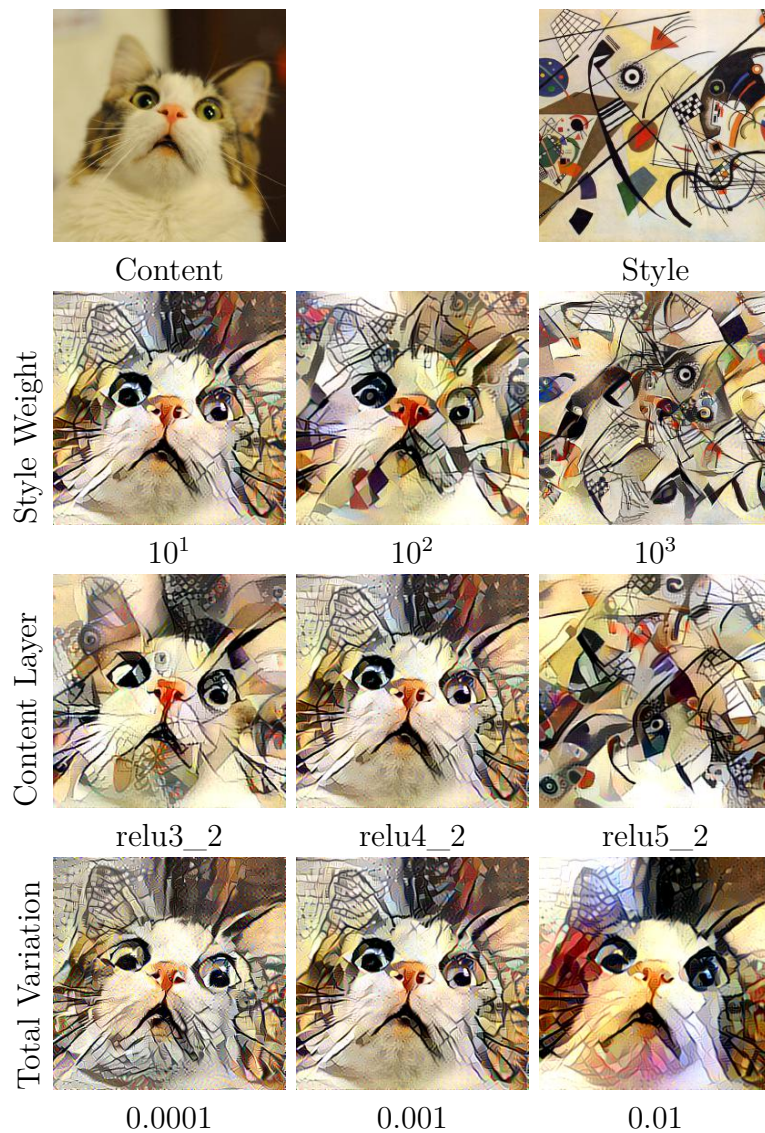


Fig. A.5 Visual results of tuning Style Transfer Network hyperparameters. The two images on top are Content and Style source images respectively, others are the resulting output obtained with a particular parameters setting.

A.3 Feed-forward network and Perceptual loss

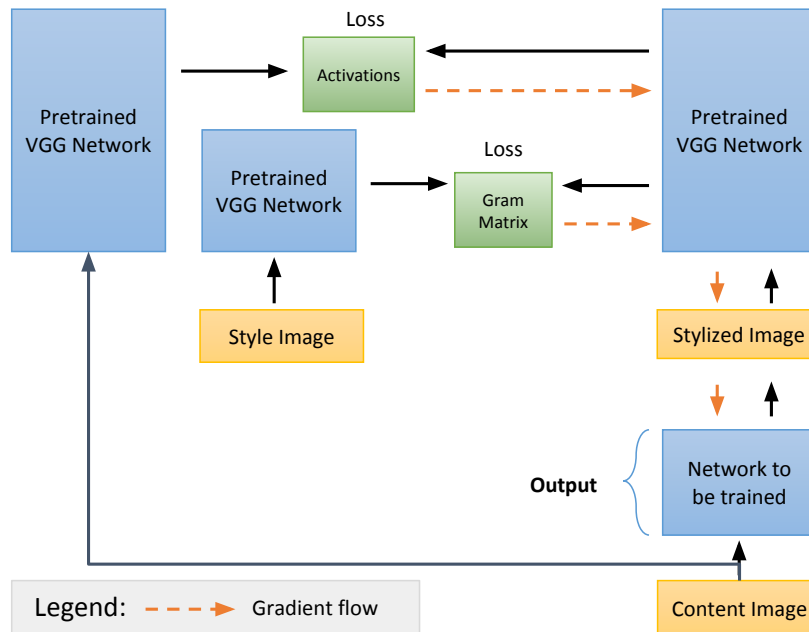


Fig. A.6

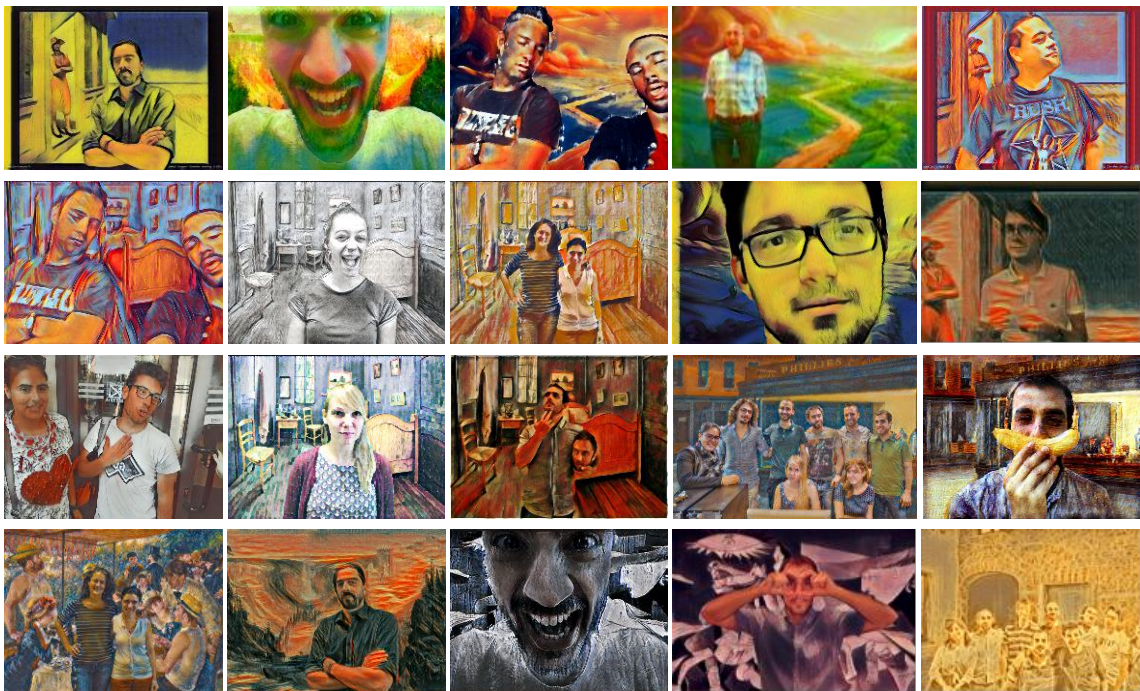


Fig. A.7 Examples of stylized images produced by the feedforward network architecture presented in Figure A.6. Images are rendered using different trained models on various styles.

