



PH.D. SCHOOL  
UNIVERSITY OF MILANO-BICOCCA

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

# Automatic Description and Annotation of Complex Scenes

Ph.D. Dissertation of: Marco Buzzelli

Supervisor: Prof. Raimondo Schettini

Co-Supervisor: Dr. Simone Bianco

Tutor: Prof. Enza Messina

Ph.D. Coordinator: Prof. Stefania Bandini

ACADEMIC YEAR 2017-2018

## Acknowledgements

I do feel like giving thanks to some people, who have been with me in different ways through the years. I thank Raimondo and Simone: they have been my guides beyond the formal roles since the very beginning, and they have often been a source for inspiration. I thank Joost van de Weijer for his invaluable guidance on illuminant estimation, during my abroad period at the Computer Vision Center. Thanks to my labmates Davide, Luigi, and Flavio: with them I had the chance to discuss and reason about anything, but most important of all I had the opportunity of being silly. I'm happy for having spent many years in such a nice group, so thanks to everybody who is or has been part of the Imaging and Vision Laboratory. Also thanks to all the lunch buddies, and thanks to my historical friends from Novara and Milano. Big thank you goes to my family: what I am now has been shaped by almost thirty years lived together, so thanks to mom, thanks to dad, and, quite incredibly, thanks to my sisters Valeria and Lucia.

But the biggest one goes to you, Gabriella. You have always been supportive and, whether you know it or not, you have helped me see the world in a different way. So thank you, you are the best.

# Abstract

Automatically describing digital images consists in extracting information that meaningfully represents the depicted elements and their attributes. The specific concept of “meaningful” can be determined by the final application: in assistance to visually impaired people, for example, the final user might want to recognize familiar elements such as landmarks and logos. In the context of driver support for smart cars, it could be useful to recognize other vehicles and pedestrians, and to tell their distance from the car itself. A general pipeline for the envisioned scenarios involves three steps: object proposal, classification, and attributes extraction. In this thesis, several methods have been studied and developed for each of these steps, and subsequently applied to specific domains with the intent of comparing the produced solutions with existing works.

Object proposal: one or many subregions containing elements of potential interest are extracted from the input image.

In this thesis, single-object proposal is achieved using a neural architecture that is optimized in a novel way, combining genetic programming for the structure optimization with back-propagation for parameters tuning.

Crossing the gap between object proposal and classification, semantic segmentation is then addressed with the definition of an original neural architecture that pays particular attention to computational efficiency for high-throughput scenarios.

Classification: the subregions generated by the object proposal phase are classified into visual classes.

Logo recognition is reported as a first case study. A new dataset has been collected, extending tenfolds the existing standard. Its combination with synthetic forms of data augmentation allows to reach state of the art performance.

Vegetables and fruits recognition is then chosen as a representative example for fine-grained visual classification problems. The task is addressed by preprocessing images with object proposal algorithms, and by exploiting the hierarchical structure of the depicted classes.

---

Attributes extraction: some subregions, identified as belonging to specific classes, are being associated with extra information.

For the task of illuminant estimation, an original learning strategy is proposed, that completely avoids the need for explicitly-annotated illuminant information, relying instead on alternatively-available object-class annotations.

Distance estimation is reported as a final case study. An alternative data representation is proposed, which is independent of any specific acquisition device, allowing the training of richer models for distance estimation.

The role of data and its representation emerges as a common theme throughout the whole thesis. In particular, the following work describes the path from relying on existing manual annotations, to gradually reducing this dependency through alternative representations and learning strategies.

# Table of contents

|   |             |
|---|-------------|
| <b>List of figures</b>  | <b>viii</b> |
| <b>List of tables</b>   | <b>x</b>    |
| <b>1 Introduction</b>   | <b>1</b>    |
| 1.1 Focus of this work . . . . .                              | 2           |
| 1.2 Significant contributions, and the role of data . . . . . | 4           |
| <b>I Object proposal</b>                                      | <b>6</b>    |
| <b>2 Introduction to object proposal</b>                      | <b>7</b>    |
| <b>3 Salient object detection</b>                             | <b>9</b>    |
| 3.1 Related works . . . . .                                   | 11          |
| 3.1.1 Bottom-up vs. top-down approaches . . . . .             | 11          |
| 3.1.2 Handcrafted vs. machine learning solutions . . . . .    | 11          |
| 3.2 Preliminary investigation . . . . .                       | 12          |
| 3.2.1 Experimental setup . . . . .                            | 14          |
| 3.2.2 Continuous-valued prediction . . . . .                  | 16          |
| 3.2.3 Multiscale analysis . . . . .                           | 18          |
| 3.3 Automatic fusion of saliency algorithms . . . . .         | 20          |
| 3.3.1 Genetic programming optimization . . . . .              | 21          |
| 3.3.2 Back-propagation optimization . . . . .                 | 23          |
| 3.4 Experiments . . . . .                                     | 27          |
| 3.4.1 Setup . . . . .   | 28          |
| 3.4.2 Optimization results . . . . .                          | 28          |
| 3.4.3 Extension to other datasets . . . . .                   | 33          |

|           |  |           |
|-----------|--|-----------|
| <b>4</b>  | <b>Semantic segmentation</b>                         | <b>36</b> |
| 4.1       | Related works . . . . .                              | 37        |
| 4.1.1     | Accuracy-oriented architectures . . . . .            | 37        |
| 4.1.2     | Efficiency-oriented architectures . . . . .          | 38        |
| 4.2       | Dataset and evaluation metrics . . . . .             | 38        |
| 4.3       | Network design . . . . .                             | 39        |
| 4.3.1     | Evaluation setup . . . . .                           | 39        |
| 4.3.2     | Input downsampling . . . . .                         | 40        |
| 4.3.3     | Multiresolution encoder . . . . .                    | 40        |
| 4.3.4     | Fusion module . . . . .                              | 41        |
| 4.3.5     | Data augmentation . . . . .                          | 43        |
| 4.3.6     | Final architecture . . . . .                         | 43        |
| 4.4       | Comparison with other methods . . . . .              | 45        |
| <b>II</b> | <b>Classification</b>                                | <b>48</b> |
| 5         | Introduction to classification                       | 49        |
| 6         | Classification in cluttered scenes: Logos            | 52        |
| 6.1       | Preliminary investigation . . . . .                  | 53        |
| 6.1.1     | Existing works . . . . .                             | 53        |
| 6.1.2     | FlickrLogos-32 dataset . . . . .                     | 53        |
| 6.1.3     | Selective search . . . . .                           | 54        |
| 6.1.4     | Deep-feature classification . . . . .                | 55        |
| 6.1.5     | Preliminary results . . . . .                        | 56        |
| 6.2       | Proposed approach . . . . .                          | 59        |
| 6.2.1     | Logos-32plus dataset . . . . .                       | 59        |
| 6.2.2     | Shallower network . . . . .                          | 63        |
| 6.2.3     | Smarter training strategies . . . . .                | 63        |
| 6.3       | Results . . . . .                                    | 65        |
| 7         | Main-subject classification: Vegetables and fruits   | 71        |
| 7.1       | Datasets and related works . . . . .                 | 72        |
| 7.2       | Outline of vegetable and fruit recognition . . . . . | 74        |
| 7.2.1     | Neural baseline architectures . . . . .              | 74        |
| 7.2.2     | Hierarchical labeling . . . . .                      | 74        |
| 7.2.3     | Saliency-based proposal . . . . .                    | 75        |

|            |   |            |
|------------|---|------------|
| 7.3        | Experiments . . . . .   | 75         |
| <b>III</b> | <b>Attributes extraction</b>                                      | <b>80</b>  |
| <b>8</b>   | <b>Introduction to attributes extraction</b>                      | <b>81</b>  |
| <b>9</b>   | <b>Illuminant estimation</b>                                      | <b>83</b>  |
| 9.1        | Related works . . . . .   | 84         |
| 9.2        | Proposed method for illuminant estimation . . . . .               | 85         |
| 9.2.1      | Object Recognition network . . . . .                              | 85         |
| 9.2.2      | Illuminant Estimation network . . . . .                           | 86         |
| 9.3        | Experiments . . . . .   | 87         |
| 9.3.1      | Experimental setup . . . . .                                      | 87         |
| 9.3.2      | Input normalization . . . . .                                     | 88         |
| 9.3.3      | Results . . . . .   | 91         |
| 9.3.4      | Analysis on the training-set color bias . . . . .                 | 92         |
| <b>10</b>  | <b>Distance estimation</b>  | <b>93</b>  |
| 10.1       | Background and related works . . . . .                            | 95         |
| 10.2       | Proposed representation for distance information . . . . .        | 97         |
| 10.3       | Datasets transformation to common representation . . . . .        | 100        |
| 10.4       | Experiments . . . . .   | 103        |
| 10.4.1     | Training preprocessing . . . . .                                  | 104        |
| 10.4.2     | Evaluation procedure . . . . .                                    | 105        |
| 10.4.3     | Experimental results . . . . .                                    | 106        |
| 10.4.4     | Applicability of the proposed representation to other methods . . | 107        |
| 10.4.5     | Multi-class evaluation . . . . .                                  | 109        |
| <b>11</b>  | <b>Conclusions</b>  | <b>113</b> |
|            | <b>References</b>   | <b>116</b> |

# List of figures

|     |  |    |
|-----|--|----|
| 1.1 | Stages of the proposed pipeline for image description and annotation. . . . .      | 2  |
| 2.1 | Possible scenarios for the object proposal phase. . . . .                          | 7  |
| 3.1 | Difference in saliency annotation strategy for three datasets. . . . .             | 10 |
| 3.2 | Activations of the Fully Convolutional Network for saliency estimation. . . . .    | 13 |
| 3.3 | ROC curves derived from continuous-valued and binary prediction. . . . .           | 17 |
| 3.4 | Schematic view of multiscale analysis at training and inference time. . . . .      | 18 |
| 3.5 | Average of the three $F_\beta$ variants on different datasets. . . . .             | 20 |
| 3.6 | Effect of multiscale analysis on example images. . . . .                           | 21 |
| 3.7 | Fusion trees generated through Genetic Programming. . . . .                        | 30 |
| 3.8 | Different implementation stages of the automatically designed DL-MAE . . . . .     | 32 |
| 3.9 | Visual results of the extended CNN from DL-MAE . . . . .                           | 35 |
| 4.1 | Simplified representation of the architecture for semantic segmentation. . . . .   | 39 |
| 4.2 | The multiresolution network used for semantic segmentation. . . . .                | 42 |
| 4.3 | Experimented fusion modules. . . . .   | 42 |
| 4.4 | Plot of mIoU vs FPS for comparison with the state of the art. . . . .              | 46 |
| 4.5 | Semantic segmentation results obtained with the proposed architecture. . . . .     | 47 |
| 5.1 | Different approaches for the classification phase. . . . .                         | 49 |
| 6.1 | Sample images from the FlickrLogos-32 dataset. . . . .                             | 54 |
| 6.2 | Partial representation synthetic data augmentation. . . . .                        | 57 |
| 6.3 | Preliminary results on the Flickr-Logos dataset. . . . .                           | 58 |
| 6.4 | Graphical comparison between FlickrLogos-32 and Logos-32plus dataset. . . . .      | 60 |
| 6.5 | Simple query results and extended query results. . . . .                           | 61 |
| 6.6 | Structure of shared keywords among different logo classes. . . . .                 | 61 |
| 6.7 | Example of near-duplicates detected by the developed procedure. . . . .            | 62 |
| 6.8 | Visualization of the contribution of real and synthetic data augmentation. . . . . | 65 |

|      |   |     |
|------|---|-----|
| 6.9  | Detailed training procedure developed for logo recognition. . . . .               | 65  |
| 6.10 | Complete testing procedure applied for logo recognition. . . . .                  | 66  |
| 6.11 | Examples of wrongly-labeled logos. . . . .  | 70  |
| 7.1  | Some of the challenges specific to fine-grained visual classification. . . . .    | 71  |
| 7.2  | Detail of the hierarchical annotation provided with the VegFru dataset. . . . .   | 73  |
| 7.3  | Two fully-connected specializations to exploit hierarchical labels. . . . .       | 76  |
| 7.4  | Often-confused class pairs for the best performing model. . . . .                 | 77  |
| 7.5  | Saliency-driven object proposal for example images in the VegFru dataset. . . . . | 79  |
| 8.1  | Range of possibilities for the attributes extraction phase. . . . .               | 81  |
| 9.1  | Schematic representation of the proposed learning strategy. . . . .               | 85  |
| 9.2  | Samples from VegFru dataset, showing the importance of color. . . . .             | 88  |
| 9.3  | Complete pre- and post-processing for the proposed strategy. . . . .              | 90  |
| 9.4  | Example color corrections of the proposed method. . . . .                         | 92  |
| 10.1 | Importance of camera parameters in distance estimation. . . . .                   | 95  |
| 10.2 | Visualization of the proposed size-based representation of distance. . . . .      | 98  |
| 10.3 | Mask generation and ground truth preprocessing. . . . .                           | 100 |
| 10.4 | Fully Convolutional Network employed for distance estimation. . . . .             | 101 |
| 10.5 | Difference in content and format among the used training data. . . . .            | 102 |
| 10.6 | Distribution of <i>real_size</i> values of the four datasets. . . . .             | 105 |
| 10.7 | Example per-pixel estimation on a test image. . . . .                             | 108 |
| 10.8 | Example predictions on the people class. . . . .                                  | 111 |
| 10.9 | Example predictions on all analyzed semantic classes. . . . .                     | 112 |
| 11.1 | Highlighted uses cases in the proposed pipeline. . . . .                          | 113 |

# List of tables

|      |  |    |
|------|--|----|
| 3.1  | Details of the adopted fully convolutional architecture. . . . .                     | 14 |
| 3.2  | Summary of tested datasets for saliency estimation. . . . .                          | 15 |
| 3.3  | Comparison of performance for binary and continuous-valued estimation. . . . .       | 16 |
| 3.4  | Evaluation results for different input resolutions and combinations. . . . .         | 19 |
| 3.5  | The set of functional symbols used in genetic programming. . . . .                   | 22 |
| 3.6  | Comparison of traditional and continuous statistics computation. . . . .             | 25 |
| 3.7  | Availability of input saliency algorithms on various datasets. . . . .               | 29 |
| 3.8  | Results for fusion trees at different levels of optimization. . . . .                | 31 |
| 3.9  | Comparison of the optimized fusion trees with input saliency algorithms. . . . .     | 33 |
| 3.10 | Comparison of deep learning fusion trees and input saliency algorithms. . . . .      | 34 |
|      |  |    |
| 4.1  | Impact of downsampling on the system performance. . . . .                            | 40 |
| 4.2  | Performance and speed on Cityscapes validation set for different encoders. . . . .   | 41 |
| 4.3  | mIoU and FPS on Cityscapes validation set and for different fusion modules. . . . .  | 43 |
| 4.4  | mIoU on Cityscapes validation set with different forms of data augmentation. . . . . | 43 |
| 4.5  | The proposed network architecture. . . . .   | 44 |
| 4.6  | Comparison with state-of-the-art methods on Cityscapes test set. . . . .             | 46 |
|      |  |    |
| 6.1  | Official partitions of the FlickrLogos-32 dataset. . . . .                           | 58 |
| 6.2  | Performance comparison with other approaches. . . . .                                | 58 |
| 6.3  | Details comparison of FlickrLogos-32 and Logos-32plus datasets. . . . .              | 60 |
| 6.4  | Architecture of the shallower neural network used for logo classification. . . . .   | 63 |
| 6.5  | Analysis on the contribution of training choices on global performance. . . . .      | 67 |
| 6.6  | Best training configuration compared with methods in the state of the art. . . . .   | 69 |
|      |  |    |
| 7.1  | Results for baseline architectures on both the super- and sub-class problem. . . . . | 76 |
| 7.2  | Results for hierarchical labeling exploitation on the sub-classes problem. . . . .   | 78 |
| 7.3  | Impact of saliency-driven object proposal on tested neural architectures. . . . .    | 78 |

|      |   |     |
|------|---|-----|
| 9.1  | Performance of different algorithms for illuminant estimation. . . . .    | 89  |
| 10.1 | Cardinality of the chosen datasets. . . . .                               | 103 |
| 10.2 | Error measures on CityScapes validation set for the “people” class. . . . | 106 |
| 10.3 | Errors obtained on different object classes. . . . .                      | 110 |

# Chapter 1

## Introduction

*“Can you tell me what it means? [...] Describe the vision, the meaning is missing.”*

P.O.D. - Sleeping Awake

From the retina, through the optic nerve, and finally reaching the visual cortex of the brain. The human vision system takes place at different levels of this path [165], allowing people to process the visual information that comes from their environment and to effectively make sense of it. This form of “understanding” of the visual input played a crucial role in the survival of the species, and eventually contributed to more abstract functions, such as the contemplation of beauty in arts and nature [61].

Providing computers with the ability to understand the content of a digital image at the same level of human beings is considered to be an ambitious and, currently, distant objective [87]. As each simpler problem is solved, in fact, it is only natural to pose a more complex one, leading to questions that require complex inference and reasoning in order to provide an adequate answer [86]. Human-level understanding could then be, rather than a precise goal, a general compass in conducting scientific research, providing global drive and direction. Further motivation may then come from practical and concrete applications: assisting visually-impaired people through object recognition is an excellent example of a feasible and powerful use of computer vision. Similar techniques can also be applied to more-commercial uses, such as automatic photo and video tagging of personal collections. Other forms of image analysis are then involved in the development of smart cars, where a typical requirement would be to infer the distance between the vehicle and various items on the road [13].

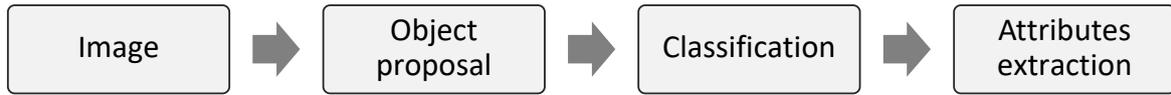


Fig. 1.1 Stages of the proposed pipeline for automatic image description and annotation.

## 1.1 Focus of this work

The subject of this thesis is the automatic description and annotation of complex scenes. A complex scene can be defined as an image depicting multiple elements [176]. The final application will typically determine whether all these subjects should be analyzed, or if only one is the main object of interest (with the others considered as distraction elements). For example, in the context of a smart car, every piece of information that is acquired from visual sensors should be taken into consideration for a safe decision-making phase [73]. Conversely, any single-object recognition task, such as fine-grained identification of plants and vegetables, can potentially benefit from the exclusion of non-salient elements [218]. In both situations, an automatic description of these scenes is at the core of virtually any computer-vision-driven task. Specifically, the focus of this thesis will be on categorical description (“annotation”) and numerical description, i.e. assigning a value to different parts of the input image. Other types of description are possible, including for example textual information. Although not covered here, textual description of digital images shares basic techniques of middle-level feature extraction with the methods involved in this work [202], thus forming a natural extension.

This thesis follows a general pipeline for complex scene description and annotation, shown in Figure 1.1, which is composed of three main steps:

1. Object proposal
2. Classification
3. Attributes extraction

Such a division can be helpful in structuring and categorizing the addressed tasks, which will be later introduced. It should be however interpreted as a flexible guide, rather than a strict formula.

**Object proposal.** The first step consists in defining one or more subregions of the input image that are likely to contain an object of interest. As such, this plays an essential role in the analysis of complex scenes. When the input image is supposed to contain a

single element, estimating a dense saliency map can produce a useful piece of information to locate the main subject itself. Once the main object proposal is determined, the corresponding subregion can then be fed for example to a classification module. Similarly, if the image contains several objects of interest, an object proposal algorithm should produce multiple subwindows, which are then also fed to a separate classification module for further analysis. This two-step separation can be useful in reducing the overall computational effort by restricting the time-consuming classification phase to a limited amount of subwindows.

Falling in-between the concepts of proposal and classification is semantic segmentation. This is the task of automatically assigning each pixel in the image to a class, including the possibility of a background/reject class. In this alternative formulation, the two steps of proposal and classification cannot be logically separated. Other works from the literature that consider the first two steps as a unique block typically include object detection methods, such as Deformable Part Models (DPM) [68] and the You-Only-Look-Once (YOLO) architecture [161]. The separate distinction adopted in this thesis is corroborated by the work of Girshick et al. on “Regions with CNN features” (R-CNN) [82].

**Classification.** Assigning the content of a digital image (or part of it) to a categorical class is at the core of many computer vision applications. Different instances of the problem impose different challenges, as well as facilitations, on its solution. For example, recognizing planar objects such as traffic signs and brand logos allows for specific assumptions to be made on the type of distortions that can be encountered in the wild. The same assumptions cannot be made for recognition of deformable objects, such as vegetables and fruits, for which different points of view and high intra-class variability have a stronger influence on the final object appearance. In both situations, relying on a robust model largely depends on the availability of wide collections of labelled training data. Whenever this condition cannot be satisfied, transfer learning has been proven an invaluable resource for good classification performance: a neural network already trained for the solution of a task with high data availability is used as starting point for the optimization of a second neural network specific for the final task.

**Attributes extraction.** Once an object has been identified as belonging to a specific class, further analysis can be performed to associate the object itself with numerical attributes. An example is estimating the distance between the subject of interest and the camera that took the picture, without relying on stereoscopic vision. In this case,

## 1.2 Significant contributions, and the role of data

---

explicitly or implicitly understanding the semantics of depicted objects is fundamental in producing a reasonable approximation of their distance from the camera. Attributes can also be extracted by analyzing the whole image, but still exploiting semantic awareness to some extent. For instance, illuminant estimation consists in determining the chromatic properties of the light in the scene, and it can benefit from recognizing the objects depicted in the image. The estimated illuminant can then be used to color correct the image: either for aesthetic purposes or as a preprocessing step to other computer-vision-related tasks.

The structure of this thesis follows the aforementioned pipeline. Part I is dedicated to object proposal, and presents developed methods for salient object detection, and for semantic segmentation. Part II addresses classification problems, including logo recognition, and vegetable recognition. Part III covers attributes extraction, specifically illuminant estimation and distance estimation.

## 1.2 Significant contributions, and the role of data

The general-purpose methods here studied and developed, are applied to specific applications with the intent of comparing the produced work with existing solutions.

The thesis begins with salient object detection, an ill-posed problem first addressed by heavily relying on manually-annotated images, in order to have a universal concept of “saliency” to naturally emerge from the data itself. The same problem is then reformulated as recombining the saliency maps generated by methods existing in literature. This is achieved using a neural architecture that is optimized in a novel way, combining genetic programming for the structure optimization with backpropagation for parameters tuning. This formulation is a first step towards changing the nature of training data, as the proposed solution takes as input the predictions of existing methods instead of the original images.

Semantic segmentation is introduced as the crossing bridge between object proposal and classification. The focus is put on segmentation of street scenes, exploiting a standard dataset that depicts pedestrians, cars, buildings, and traffic signs. By envisioning a smart-car application scenario, high emphasis is given on the development of an original model for semantic segmentation that is highly competitive in terms of accuracy/speed trade-off.

The focus is later shifted to classification tasks: for logo recognition, the available datasets were not deemed sufficient, therefore it was felt appropriate to build a new logos

## 1.2 Significant contributions, and the role of data

---

dataset, extending tenfolds the existing standard. The combination of a new dataset with synthetic forms of data augmentation allows for state of the art performance in logo recognition to be reached.

Vegetable and fruit recognition is selected as a case study for fine-grained visual classification. The task is initially addressed with transfer learning, leveraging on the discriminative power obtained by pre-training on a different type of input data. Then, the hierarchical nature of annotated classes is exploited with two proposed techniques, in order to improve the overall recognition performance.

For the task of illuminant estimation, an original learning strategy is proposed, that completely avoids the need for explicitly-annotated illuminant information, relying instead on alternatively-available object-class annotations. Specifically, recognition of vegetable classes is used as an auxiliary task to latently train a color constancy neural network, which is made possible thanks to the importance of color in vegetable discrimination.

Finally, for distance estimation, an alternative data representation is proposed, which is independent from any specific acquisition device. In this alternative description, each pixel of an image is associated with the size, in real life, of what it represents. Datasets acquired with different devices can therefore be effortlessly combined to build more powerful models, and monocular distance estimation can be performed on images acquired from new devices as well.

A common theme throughout this thesis is the role of data in the development of computer vision systems, a crucial aspect in the field of machine learning. What emerges from this work is not only the importance of training data itself, but also the path to gradually becoming less dependent on it.

# Part I

## Object proposal

# Chapter 2

## Introduction to object proposal

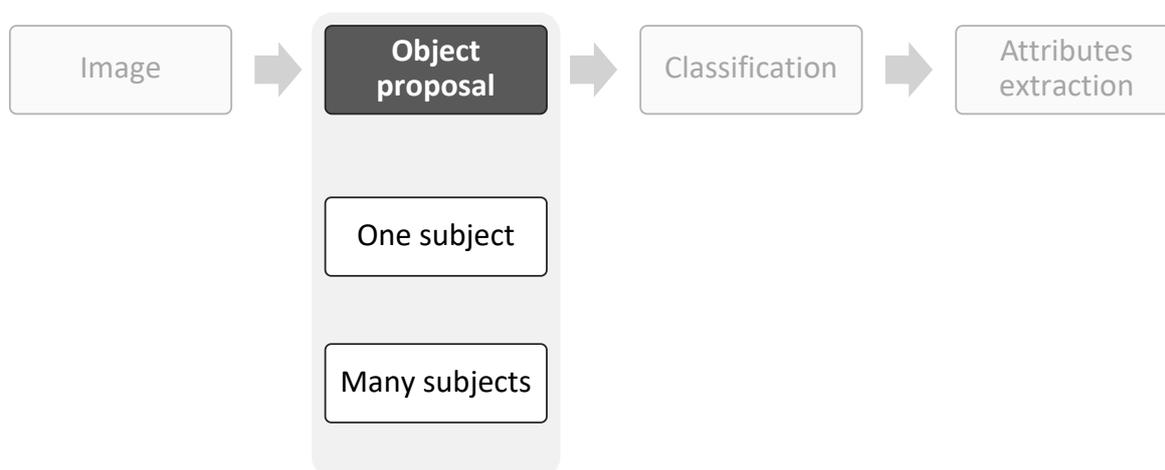


Fig. 2.1 Possible scenarios for the object proposal phase.

Following the general pipeline outlined in Chapter 1, the very first step in image description is the generation of object proposals, i.e. subregions of the input image that are likely to depict meaningful content for a given subsequent task. A complex scene is by definition characterized by the presence of multiple subjects. Depending on the final application, it is then possible to describe two main scenarios, as shown in Figure 2.1:

- Only one object is the subject of interest, and the others should be considered as distraction elements. In this case, an estimated saliency map can locate the main subject, and focus the attention of a subsequent classification module, eventually improving recognition performance.
- Every object is potentially interesting and should be considered for further analysis. For multiple subjects, the explicit separation between proposal and classification

---

phases can reduce the computational burden by restricting the second step to a smaller set of inputs. An alternative solution consists in semantic segmentation, which encompasses both object proposal and classification at pixel-level in a unique step, by exploiting efficient ad-hoc structures, as seen in the following.

The use case of one main object of interest has been approached with the development of a general-purpose saliency estimation method [17, 18, 14]. In particular, the problem has been first analyzed with the objective of providing a visual understanding of the ill-defined concept of “salient object”. This is achieved by training a Fully Convolutional Network (FCN) on multiple saliency-related datasets, which had been annotated with drastically different criteria. Then, existing methods for salient object detection have been combined with the intent of producing a better final estimation of image saliency. This is done with a novel strategy that exploits genetic programming to automatically define a backbone CNN, which is then extended and optimized through backpropagation.

The case of multiple objects of interest has been treated in the specialized domain of semantic street segmentation [141]. This task consists in assigning to each pixel of the input image a classification label, covering street-related categories plus a background class. Such formulation effectively puts semantic segmentation in-between the steps of object proposal and classification. In this context, an encoder-decoder architecture has been developed, paying particular attention in reaching a proper balance between accuracy and speed. For a possible smart-car scenario, in fact, both elements are considered fundamental requirements.

The developed solutions for both single-object and multiple-object proposal are based on the use of deep learning techniques. Specifically, on Convolutional Neural Networks designed to produce an output that has a one-to-one spatial correspondence with the input. For saliency estimation, this means producing a dense map that represents the probability of each pixel belonging to the foreground. For semantic segmentation, multiple class-specific probability maps are produced. Selecting the arg-max for each pixel across different maps allows the production of hard semantic labels with arbitrary shapes.

Alternative techniques for object proposal, that do not rely on machine learning, involve the use of hand-crafted algorithms and features. Many solutions belonging to this category [97] approach the inherent ambiguity of class-agnostic image segmentation by producing multiple overlapping object proposals, therefore aiming at a high recall at the expense of precision. This is the case for the selective search algorithm [191], which has been adopted and tuned as the object proposal phase for logo recognition, as described in Chapter 6.

# Chapter 3

## Salient object detection

*“Che cosa ti sfugge quando sei certo di aver guardato già dappertutto?  
[...] Non è il senso, infatti non c’è niente di semantico.”*

Uochi Toki - I fonici

The human vision system is able to efficiently detect salient areas in a scene. These areas are then further processed by the brain to extract high-level information [179, 175], a mechanism that inspired the “object proposal - classification – attributes extraction” steps in the proposed pipeline for image annotation. Visual saliency has been primarily studied by neuroscientists, cognitive scientists and recently has received attention from other research communities working in the fields of computer vision, computer graphics and multimedia. The aim of these researches is to build and exploit computational visual saliency models in different application domains. In fact, visual saliency can be used for salient object detection emphasizing object-level regions in the scene that can serve as a pre-processing step in various computer vision and processing tasks. Salient object detection is useful for scene recognition [74, 163], object detection [144, 146], segmentation [122], and tracking [136]. It can also be exploited for image manipulation and visualization as in applications such as image retargeting [8], image collage [138], and non-photorealistic rendering [59]. Moreover, in multimedia application saliency can be exploited for image and video summarization [151, 54, 88], retrieval [77], and image quality or aesthetic assessment [118, 199].

Despite the clear advantage that would be gained from solving this task, there is no universally-accepted definition on what makes an element “salient”, thus rendering saliency estimation particularly challenging. This can be better seen by observing Figure 3.1: while the main object of interest in the first image can be generally recognized as the butterfly itself, the other two examples present less obvious answers. Figure 3.1(b)



Fig. 3.1 Difference in saliency annotation strategy for three datasets: THUR15K [45] (a), PASCAL-S [125] (b), and JuddDB [30] (c).

shows a crowded dining scene with no clear main subject. The annotators of the corresponding dataset [125] addressed this problem by assigning a decreasing level of saliency to each segmented element in the images. Note that this saliency rank was computed by collecting gaze data from multiple observers. In a similar fashion, Figure 3.1(c) provides another non-trivial example, annotated in the corresponding dataset [30] with only the most looked-at region, according to human observers.

Both the extreme subjectivity intrinsic to the annotation task, and the criteria heterogeneity adopted by different dataset curators, contribute to making even more difficult a problem that is ill-posed in the first place. Methods for saliency estimation that are based on handcrafted low-level features have always struggled in reaching good performance [31]. The observed characteristics suggest, in fact, that a data-driven model with at least some level of semantic awareness would be essential to properly address the proposed task. A preliminary investigation has been therefore here conducted, with the specific objective of having a universal concept of image saliency to naturally emerge from a large set of heterogeneously-annotated data.

In order to make the detection more robust and to improve the generalization capabilities, then, saliency methods often integrate different features [130] or fuse saliency maps generated from different methods [137]. However, the features and the combination strategies are usually empirically designed, i.e. which features to use and how to combine them remains an open problem. To this extent, the second part of this chapter is focused on the design of a novel saliency detection approach by leveraging the outputs of existing saliency detection methods in a supervised way. In particular, a candidate solution for combining the saliency maps is first created using genetic programming (GP) and a set of provided symbols. The solution and some of the involved operations have parameters

that cannot be easily (or efficiently) optimized within the GP framework, and they are therefore used as a blueprint upon which designing the architecture of an equivalent Convolutional Neural Network (CNN). Within the CNN optimization framework, based on back-propagation, it is easier to search for the optimal parameters of the GP solution. Moreover, variants can also be easily created by including additional operations on intermediate results that can be either ignored or optimized by the CNN.

## 3.1 Related works

### 3.1.1 Bottom-up vs. top-down approaches

Traditionally, saliency detection methods can be divided into two categories: bottom-up and top-down saliency detection. Bottom-up saliency methods are stimuli-driven [103]. The saliency is usually modeled by local or global contrast on hand-crafted visual features, and knowledge about human visual attention is embedded in the model exploiting some heuristic priors such as texture [6], background [198], compactness [154], and objectness [125]. With these methods, no explicit information about the semantics of the salient regions is provided, but it is indirectly embedded via prior assumptions that are made on the location, shape or visual properties of the salient regions to be detected. Bottom-up methods can be considered generic in the sense they can be used in different contexts as general purposed approaches.

Top-down saliency methods, on the contrary, are designed to find regions in the images that are relevant for a given task. They are often also referred to as “task-driven approaches”. These methods usually formulate the saliency detection as a supervised learning problem [106]. The rationale of top-down saliency methods is to identify image regions that belong to a pre-defined object category [48]. For this reason, these methods are theoretically more robust for identifying salient regions in cluttered backgrounds where bottom-up methods fail due to them relying on visual stimuli and not on semantics. Since top-down approaches rely on the use of some form of training data to build the detection model, they can instead be very robust.

### 3.1.2 Handcrafted vs. machine learning solutions

An alternative categorization of saliency estimation methods can be based on whether they rely on handcrafted features, or on a machine learning approach. It can be observed, however, that handcrafted solutions generally rely on a bottom-up logic [31], thus producing a partial overlap with the original categorization.

The method presented in Discriminative Regional Feature Integration (DRFI) [106] builds a multi-level representation of the input image, and creates a regression model mapping the regional feature vector of each level to the corresponding saliency score. These scores are finally fused in order to determine the complete saliency map. In Quantum Cut (QCUT) [9] authors model salient object segmentation as an optimization problem. They then exploit the link between quantum mechanics and graph-cuts to develop an object segmentation method based on the ground state solution of a modified Hamiltonian. The authors of Minimum Barrier Distance (MBD) [210] present an approximation of the MBD transform, and combine it with an appearance-based “backgroundness” cue. The resulting method performs significantly better than other solutions having the same computational requirements. In Saliency Tree (ST) [131] authors simplify the image into primitive regions, with associated saliency based on multiple handcrafted measures. They generate a saliency tree using region merging, and perform a systematic analysis of such tree to derive the final saliency map. Robust Background Detection (RBD) [219] introduces boundary connectivity: a background measure based on an intuitive geometrical interpretation. This measure is then used along with multiple low-level cues to produce saliency maps through a principled optimization framework.

Recently Convolutional Neural Networks (CNNs) approaches are being used to automatically learn features in a data-driven way [121, 98, 18]. CNNs-based approaches mimic well the Human Visual System in that they are able to process visual information at different levels of details and semantics. This characteristic makes them a very powerful tool for learning feature representations in different application domains [177, 47, 38, 51]. Recurrent Fully Convolutional Networks (RFCN) [194] adopt a different strategy built on top of the concept of FCNs, integrating it with backward self-correcting connections as well as saliency prior knowledge. Multi-Context deep learning (MC) [215], one of the first solutions to address saliency estimation with the use of convolutional neural networks, defines a unified framework to represent both global and local context in a data-driven fashion. The authors of Deeply Supervised Saliency (DSS) [99] introduce short connections to the skip-layer structures described by the Holistically-Nested Edge Detector architecture [201], providing an alternative way to generate rich multiscale feature maps at different layers.

## 3.2 Preliminary investigation

This preliminary study aims at providing an intuitive understanding of the concept of “saliency”, based on the consensus of different datasets annotation criteria, and at

## 3.2 Preliminary investigation

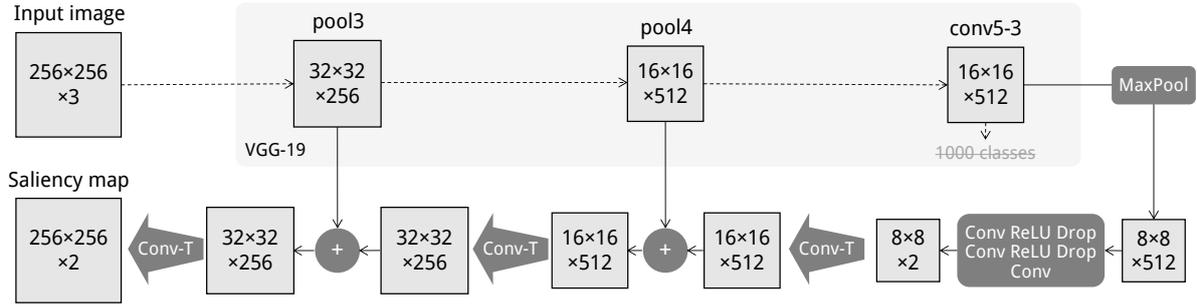


Fig. 3.2 Activations of the Fully Convolutional Network employed for saliency estimation, with an input example of resolution  $256 \times 256$  pixels. Details of the VGG-19 network [181] are omitted for ease of visualization.

acquiring sensibility over the behavior of different evaluation measures. This goal is here obtained through a Fully Convolutional Neural Network (FCN [133]) that exploits layers previously trained on recognizing 1000 object classes [173] as the starting point for a deep analysis of the original input image, in order to produce a per-pixel estimation of its saliency. The resulting architecture, after being properly trained, is able to generate an estimation of object saliency that transcends the 1000 classes defined for the pre-training. These classes are, in fact, used to build a semantically-aware internal representation, but do not constrain the type of objects that can be identified as being “salient”. A simple proof of this is the “person” category, which is absent from the original set of classes, but well prominent in the final saliency estimation results, as shown in Section 3.2.1. The overall structure of the proposed architecture is shown in Figure 3.2, while details are provided in Table 3.1: the output of layer *conv5-3* from a VGG-19 network (Visual Geometry Group [181]) is mapped to the final problem size (i.e. two channels, for “salient” and “non-salient”) by using a series of Pooling, Convolution, ReLU, and DropOut blocks. The result is then combined with the outputs of *pool4* and *pool3* by direct sum. Since these activations all have a different spatial resolution, two convolutional-transpose layers (also known as fractionally-strided convolutions) are used to bring them to a compatible size, and a third one is used to map the result to the original input size. The whole network is trained end-to-end, eventually updating also the pre-learned weights that were used to initialize the VGG-19 module. This solution also involves continuous-valued prediction and multiscale analysis, which are experimentally proven to positively impact the accuracy of the saliency estimation.

## 3.2 Preliminary investigation

Table 3.1 Details of the adopted fully convolutional architecture, with an input example of resolution  $256 \times 256$  pixels. Layers marked with [square brackets] come from the original VGG-19 network [181], whose details are omitted for ease of visualization.

| Operation | Input layers    | Output layer | Filter size    |       |        | Stride | Output size               |
|-----------|-----------------|--------------|----------------|-------|--------|--------|---------------------------|
|           |                 |              | Kernel         | Ch.in | Ch.out |        |                           |
| MaxPool   | [conv5-3]       | pool5        | $2 \times 2$   | -     | -      | 2      | $8 \times 8 \times 512$   |
| Conv      | pool5           | conv6        | $7 \times 7$   | 512   | 4096   | 1      | $8 \times 8 \times 4096$  |
| ReLU      | conv6           | relu6        | -              | -     | -      | -      | $8 \times 8 \times 4096$  |
| DropOut   | relu6           | drop6        | -              | -     | -      | -      | $8 \times 8 \times 4096$  |
| Conv      | drop6           | conv7        | $1 \times 1$   | 4096  | 4096   | 1      | $8 \times 8 \times 4096$  |
| ReLU      | conv7           | relu7        | -              | -     | -      | -      | $8 \times 8 \times 4096$  |
| DropOut   | relu7           | drop7        | -              | -     | -      | -      | $8 \times 8 \times 4096$  |
| Conv      | drop7           | conv8        | $1 \times 1$   | 4096  | 2      | 1      | $8 \times 8 \times 2$     |
| Conv-T    | conv8           | convT1       | $4 \times 4$   | 2     | 512    | 1/2    | $16 \times 16 \times 512$ |
| Sum       | [pool4], convT1 | sum1         | -              | -     | -      | -      | $16 \times 16 \times 512$ |
| Conv-T    | sum1            | convT2       | $4 \times 4$   | 512   | 256    | 1/2    | $32 \times 32 \times 256$ |
| Sum       | [pool3], convT2 | sum2         | -              | -     | -      | -      | $32 \times 32 \times 256$ |
| Conv-T    | sum2            | convT3       | $16 \times 16$ | 256   | 2      | 1/8    | $256 \times 256 \times 2$ |

### 3.2.1 Experimental setup

These experiments have been designed to quantify the practical contribution of each individual element of the proposed method for preliminary investigation of saliency estimation, in particular: continuous-valued prediction and multiscale analysis at inference time. All conducted tests follow the benchmark proposed in Reference 31 in terms of datasets and evaluation measures.

The seven tested datasets, presented in Table 3.2, offer different types of images and are annotated with sometimes drastically different criteria, as previously noted. For the purpose of these experiments, a Leave-One-Dataset-Out (LODO) setup has been adopted, as the original benchmark does not provide an official training-test split for each dataset. This solution guarantees overfitting-free results, as the model is never tested on the same annotation criteria that are used during the training phase. Cross-dataset near-duplicate removal is also conducted to avoid the same images being present at both training and test time in the adopted LODO setup.

Different metrics have been used to analyze different aspects of the estimated image saliency:

Table 3.2 Summary of tested datasets for saliency estimation.

| Dataset        | Images | Mean size (px) | Notes                               |
|----------------|--------|----------------|-------------------------------------|
| MSRA10K [130]  | 10000  | 400×300        | -                                   |
| THUR15K [45]   | 6233   | 450×300        | Only 6233/15000 annotated images    |
| DUTOMRON [206] | 5166   | 400×300        | -                                   |
| ECSSD [203]    | 1000   | 400×300        | -                                   |
| JuddDB [30]    | 900    | 1024×768       | Salient object typically very small |
| PASCALS [125]  | 850    | 500×350        | High background clutter             |
| SED2 [5]       | 100    | 300×250        | Two salient objects per picture     |

**F-Measure** ( $F_\beta$ ) is the weighted harmonic mean between precision and recall:

$$F_\beta = \frac{(1 + \beta^2)Precision \times Recall}{\beta^2 Precision + Recall} \quad (3.1)$$

In order to give more weight to precision, which is considered to be more important than recall for this task [1, 130, 31], parameter  $\beta^2$  is set to 0.3. The continuous-valued saliency estimation can be binarized with different techniques before effectively computing precision and recall. The adopted benchmark presents three alternative ways to perform such binarization:

1. Varying fixed threshold: Precision and Recall are computed at all integer thresholds between 0 and 255, and then averaged.
2. Adaptive threshold [1]: The threshold for binarization is set to twice the mean value of the predicted saliency map.
3. Saliency Cut [46]: The threshold is set to a low value, thus granting a high recall rate. Segmentation algorithm GrabCut [171] is then iteratively applied to the binarized prediction, typically producing a saliency estimation with more precise edges.

**Area Under Curve (AUC)** is the area under the Receiver Operating Characteristic curve. The ROC curve is in turn computed by varying the binarization threshold and plotting True Positive Rate (TPR) versus False Positive Rate (FPR) values:

$$TPR = \frac{TP}{TP + FN} \quad (3.2)$$

$$FPR = \frac{FP}{FP + TN} \quad (3.3)$$

## 3.2 Preliminary investigation

Table 3.3 Comparison of performance for binary estimation and continuous-valued estimation, on all considered datasets (P, T, J, D, S, M, E). Cross-dataset average is also reported. For all measures, except MAE, a higher value is better.

| Measure               | Method     | P [125] | T [45] | J [30] | D [206] | S [5] | M [130] | E [203] | Average |
|-----------------------|------------|---------|--------|--------|---------|-------|---------|---------|---------|
| $F_\beta$<br>Varying  | Binary     | 0.763   | 0.666  | 0.406  | 0.706   | 0.847 | 0.850   | 0.864   | 0.729   |
|                       | Continuous | 0.768   | 0.722  | 0.408  | 0.720   | 0.850 | 0.859   | 0.875   | 0.743   |
| $F_\beta$<br>Adaptive | Binary     | 0.688   | 0.620  | 0.382  | 0.678   | 0.857 | 0.833   | 0.783   | 0.692   |
|                       | Continuous | 0.685   | 0.617  | 0.380  | 0.652   | 0.853 | 0.834   | 0.776   | 0.685   |
| $F_\beta$<br>Sal Cut  | Binary     | 0.778   | 0.702  | 0.409  | 0.712   | 0.791 | 0.890   | 0.888   | 0.739   |
|                       | Continuous | 0.783   | 0.707  | 0.404  | 0.722   | 0.810 | 0.909   | 0.893   | 0.747   |
| AUC                   | Binary     | 0.820   | 0.851  | 0.680  | 0.828   | 0.844 | 0.877   | 0.896   | 0.828   |
|                       | Continuous | 0.949   | 0.956  | 0.807  | 0.950   | 0.970 | 0.971   | 0.979   | 0.940   |
| MAE                   | Binary     | 0.122   | 0.106  | 0.210  | 0.079   | 0.080 | 0.073   | 0.065   | 0.105   |
|                       | Continuous | 0.153   | 0.132  | 0.272  | 0.117   | 0.094 | 0.110   | 0.112   | 0.141   |

**Mean Absolute Error (MAE)** is computed directly on the prediction, without any binarization step, as:

$$MAE = \frac{1}{W \times H} \sum_{x=1}^W \sum_{y=1}^H |Prediction(x, y) - GroundTruth(x, y)| \quad (3.4)$$

where  $W$  and  $H$  refer respectively to image width and height.

### 3.2.2 Continuous-valued prediction

Most available datasets for saliency estimation and foreground detection are published with a binary ground truth [30, 45, 130, 203, 206]. It is therefore natural to approach the problem as a per-pixel binary classification task, so the FCN has been trained with a per-pixel softmax cross-entropy loss (the global loss of each mini-batch is computed by averaging all loss values from the single pixels involved). For datasets providing discrete annotations [125, 5] a preprocessing threshold has applied, setting to 1 all values greater than 0. At inference time it is then possible to stop the network processing right after the softmax layer, in order to effectively produce two complementary continuous maps, which respectively represent the probability of each pixel being, or not being, salient. If necessary, the saliency channel can then be binarized by applying a 0.5 threshold (equivalent to choosing the argmax between the two complementary channels), or using any of the thresholding techniques described in Section 3.2.1 for evaluation.

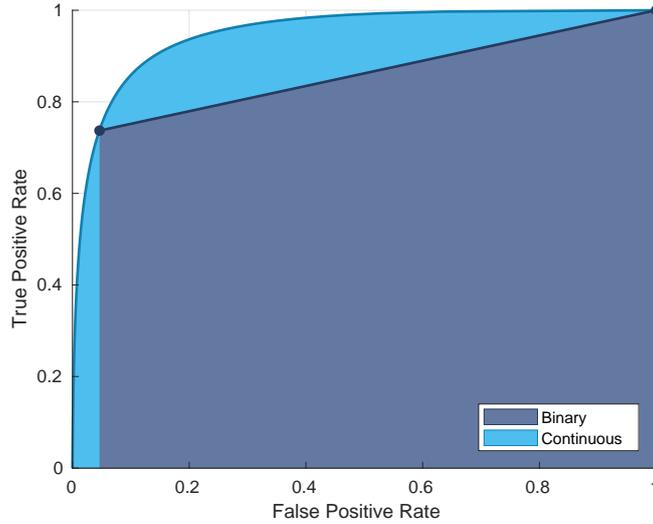


Fig. 3.3 ROC curves derived from continuous-valued and binary prediction on the DUTOMRON dataset. The graph visually shows the impact of the two solutions on computing the AUC measure (Area Under Curve).

Table 3.3 presents the comparison between continuous-valued and binary estimation, for each evaluation measure and each dataset proposed in the adopted benchmark [31]. A cross-dataset average is also provided in the last column, in order to get a global view of the impact of such contribution.

The introduction of continuous-valued estimation is mostly benefiting AUC: the area under the ROC curve. This curve is drawn by plotting the FPR-TPR value pairs obtained at each possible binarization threshold, and collapses to a single point in the degenerate case of an already-binary image. The standard evaluation procedure provided with Reference 31 generates an additional trivial solution, corresponding to an all-ones saliency estimation (FPR:1 and TPR:1). For an already-binary image, this results in a straight line between two points, and the trapezoid area underlying such line is missing two large chunks when compared to a continuous-valued evaluation, as shown in Figure 3.3, resulting in sub-optimal performance.

Conversely, Mean Absolute Error (MAE) is impacted negatively by the transition to a continuous-valued prediction. This measure is essentially a direct comparison between prediction and (binary) ground truth, so there is always going to be some residual difference on “True Positive” areas, as any continuous-valued prediction is rarely giving 100% confidence on any pixel. Such differences, however small, accumulate over the whole image and result in worse performance according to this particular evaluation measure.

Finally, the effect on  $F_\beta$  is inconsistent, but on average slightly better than what can be obtained with a binary output. It can be concluded, therefore, that producing a

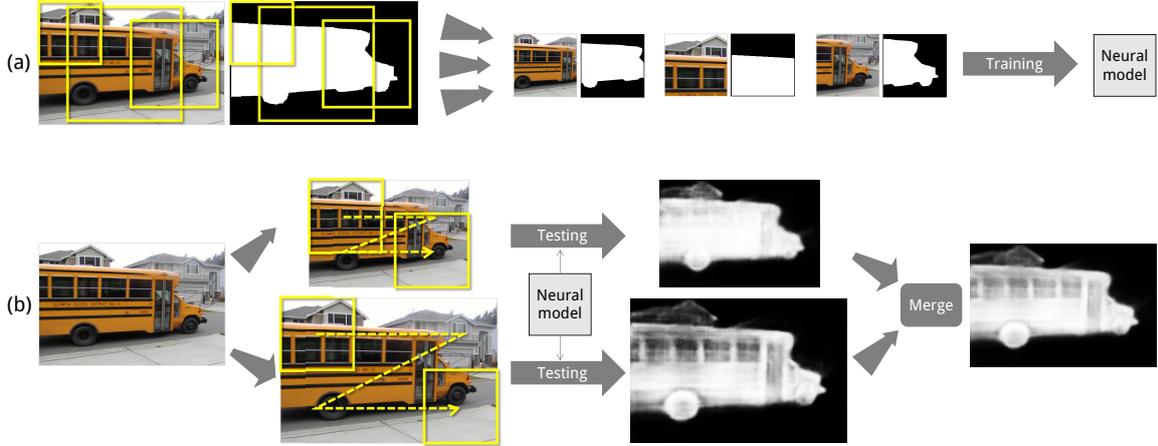


Fig. 3.4 Schematic view of multiscale analysis at training (a) and inference (b) time.

continuous-valued prediction provides a more formally correct setting for the adopted evaluation procedure, while at the same time yielding overall better results across different metrics.

### 3.2.3 Multiscale analysis

As both the input image and the portrayed elements could be of any size, it is fundamental to create a model for saliency estimation that is able to analyze the image at different scales. This kind of multiscale awareness should affect the training procedure, introducing into the learned model pieces of information that come from observing the annotated data at various scales, as well as the inference procedure, collecting saliency cues at different levels and appropriately combining them into one final output.

At training time this effect can be obtained as shown in Figure 3.4(a), by cropping subregions of random size from the input images and annotations, and eventually bringing them to a common resolution (in this case,  $256 \times 256$  pixels) in order to exploit fast mini-batch parallelization. This last resizing step will indeed destroy all information about the relationship between the crop and the rest of the image, but will work as a form of data augmentation to take into account the diversity of subject size that can be encountered at test time.

At inference time, the fully-convolutional nature of the proposed model makes it possible to process an input of any size, and consequently produce an output of the same dimensions. This, however, does not guarantee a multiscale analysis of the whole image. The neural model will, in fact, only analyze the input image on subregions of limited size (the receptive field is  $32 \times 32$  pixels), and efficiently apply this processing in a

### 3.2 Preliminary investigation

Table 3.4 Evaluation results for different input resolutions and combinations (reported values are averages across all datasets). For all measures, except MAE, a higher value is better. The configuration selected for subsequent experiments is highlighted in bold.

| Input size (px) | Merging strategy | $F_\beta$ Varying | $F_\beta$ Adaptive | $F_\beta$ Sal Cut | AUC          | MAE          |
|-----------------|------------------|-------------------|--------------------|-------------------|--------------|--------------|
| Unscaled        | -                | 0.743             | 0.685              | 0.747             | 0.940        | 0.141        |
| 256             | -                | 0.759             | 0.686              | 0.771             | 0.948        | 0.131        |
| 384             | -                | 0.773             | 0.698              | 0.772             | 0.954        | 0.124        |
| 512             | -                | 0.752             | 0.687              | 0.754             | 0.946        | 0.141        |
| 640             | -                | 0.721             | 0.671              | 0.731             | 0.931        | 0.168        |
| 768             | -                | 0.688             | 0.649              | 0.710             | 0.911        | 0.195        |
| 256, 384        | Maximum          | 0.774             | 0.677              | 0.769             | 0.955        | 0.136        |
| 384, 512        | Maximum          | 0.767             | 0.679              | 0.758             | 0.953        | 0.141        |
| 256, 512        | Maximum          | 0.769             | 0.666              | 0.759             | 0.954        | 0.149        |
| <b>256, 384</b> | <b>Average</b>   | <b>0.781</b>      | <b>0.697</b>       | <b>0.776</b>      | <b>0.957</b> | <b>0.128</b> |
| 384, 512        | Average          | 0.773             | 0.697              | 0.769             | 0.955        | 0.133        |
| 256, 512        | Average          | 0.781             | 0.697              | 0.774             | 0.958        | 0.137        |

sliding-window fashion over the whole image. In order to explicitly perform multiscale analysis, it is necessary to create copies of the input image at different resolutions (a so-called image pyramid[2]), apply the network as a sliding-window over each pyramid level, rescale all the predictions to the size of the original input, and merge the results. This procedure is shown in Figure 3.4(b).

Different scales and different merging strategies can be adopted. The first rows in Table 3.4 show how rescaling the image to one fixed size, as opposed to feeding the original image to the neural network, already brings consistently better performance among all evaluation measures (note that Table 3.4 only reports averages across all seven datasets, for reasons of readability). By picking the three most effective input sizes, i.e.  $256 \times 256$ ,  $384 \times 384$  and  $512 \times 512$  pixels, it is then possible to try different combinations. The trained model potentially assigns a high saliency score to different regions at different scales, so resizing the predictions to a common resolution, and computing a per-pixel maximum, is going to preserve such high-confidence outputs from all levels of analysis. This merging technique, however, does not produce the expected results, possibly suggesting a different level of relevance for each scale, and possibly requiring a dedicated reasoning for areas where different scales generate highly disagreeing estimations of saliency. Consistently with this last hypothesis, averaging the predictions leads to improved performance under all criteria. Also note that linear combination of independent outputs was shown to be an

### 3.3 Automatic fusion of saliency algorithms

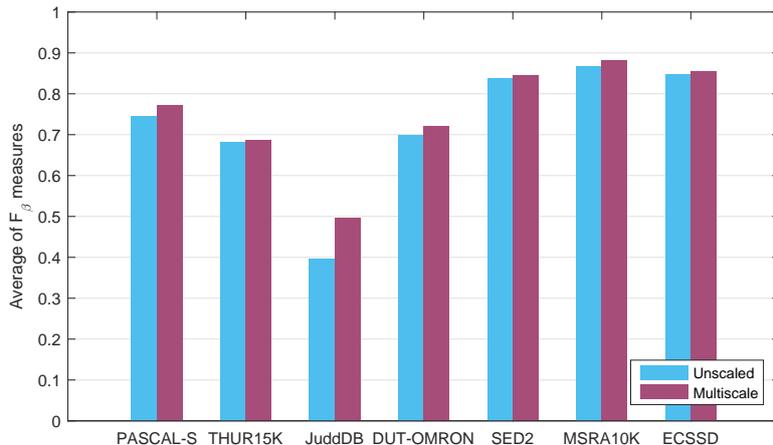


Fig. 3.5 Average of the three  $F_\beta$  variants on different datasets, for unscaled input and multiscale approach. The impact of multiscale analysis is most prominent on JuddDB dataset.

effective way of model stacking in the past [180]. Overall, the best combination consists in averaging the prediction results from 256-pixel-side images and 384-pixel-side images.

Figure 3.5 shows the impact of multiscale analysis on all seven datasets used for evaluation, reporting the average of the three  $F_\beta$  variants as a reference measure. It can be seen how dataset JuddDB[30] is the most impacted by this strategy, mainly due to its images being much larger than those of other datasets: the depicted subjects at native resolution, in fact, have a very different size with respect to the training examples. PASCALS[125] and DUTOMRON[206] are also positively affected by multiscale analysis to a meaningful, yet lower, extent.

Figure 3.6 presents three visual examples of the advantage in applying multiscale analysis. On these images, in fact, a single-scale saliency estimation would generate “holes” in the prediction due to a limited receptive field, whereas the multiscale strategy allows the model to consider the whole image. It can be observed, though, that the overall better estimation comes at the price of coarser predictions. This suggests a direction for future developments, where the fine-grained results from analysis at high resolutions might be specifically exploited to provide more accurate details.

### 3.3 Automatic fusion of saliency algorithms

The following section defines a proposed approach for the automatic design and optimization of fusion trees for existing saliency estimation algorithms. In the first phase, genetic programming techniques are exploited to combine existing saliency maps using a set of

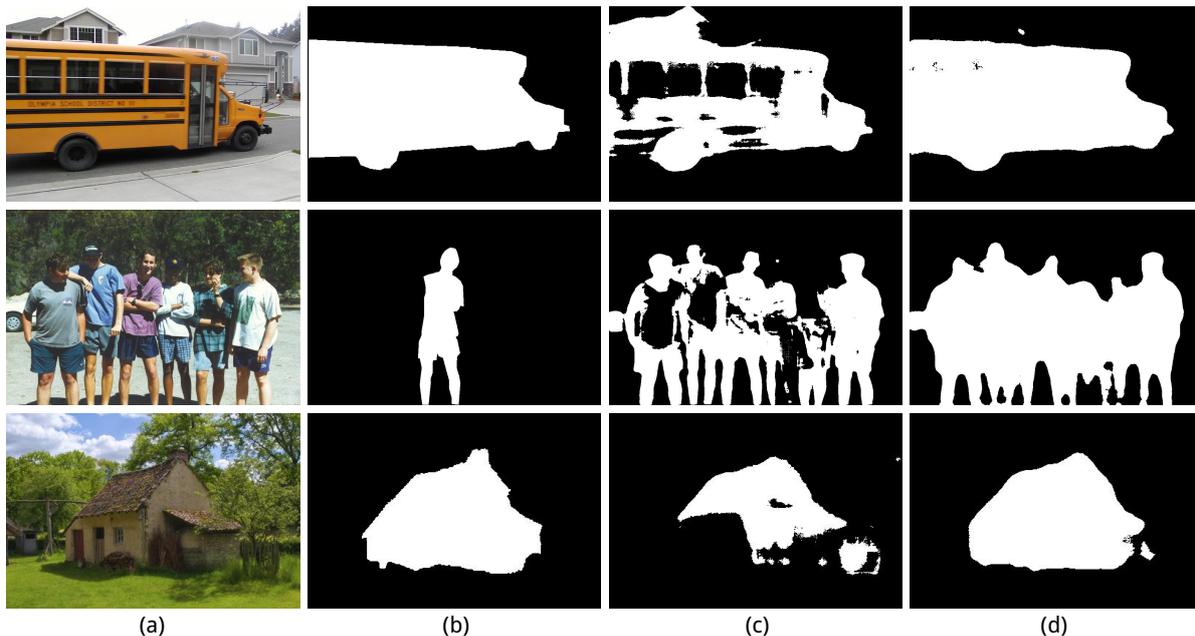


Fig. 3.6 Effect of multiscale analysis on example images: input image (a), ground truth annotation (b), single-scale saliency estimation (c), and multiscale saliency estimation (d). The content of columns (c) and (d) is here binarized to facilitate the comparison.

provided operations. The obtained fusion trees are then converted into a corresponding backbone CNN, where all operations are implemented with differentiable functions, therefore allowing the optimization of the corresponding parameters using back-propagation. Finally, the CNN is further extended with additional processing on intermediate levels of the trees, which are once again optimized through back-propagation.

### 3.3.1 Genetic programming optimization

The first step consists in finding the sequence of operations that produces the best fusion of the set of considered input saliency estimation algorithms. The candidate solutions are encoded as trees of operations, built using a set of terminal symbols  $T$  and a set of non-terminal or functional symbols  $F$ . The solution space of all the possible fusion trees is explored with Genetic Programming to find the optimal subset of input saliency estimation algorithms and the optimal sequence of operations.

Genetic programming (GP) [111] is a computational method belonging to the evolutionary computation branch of the computational intelligence research area. GP consists in the automated learning of computer programs by means of a process inspired by biological evolution. The evolution of such programs is stochastic and makes use of

### 3.3 Automatic fusion of saliency algorithms

| Symbol | Name         | Inputs   | Domain  | Effect  |
|--------|--------------|----------|---------|---|
| ERO    | Erosion      | 1        | spatial | Morphological erosion ( $5 \times 5$ square)  |
| DIL    | Dilation     | 1        | spatial | Morphological dilation ( $5 \times 5$ square) |
| MF     | Median       | 1        | spatial | 2D median filter ( $5 \times 5$ kernel)       |
| MIN    | MinStack     | $\geq 2$ | stack   | 1D minimum filter                             |
| MAX    | MaxStack     | $\geq 2$ | stack   | 1D maximum filter                             |
| MED    | MedianStack  | $\geq 2$ | stack   | 1D median filter                              |
| AVG    | AverageStack | $\geq 2$ | stack   | 1D average filter                             |
| LOG    | Quantization | 1        | pixel   | Generalized logistic non-linearity            |

Table 3.5 The set of functional symbols used in GP with their name, their  $n$ -arity, the domain on which they operate, and their corresponding operators.

genetic operators to find solutions that optimize the given objective function. The schematic search process of a generic GP algorithm is the following:

1. Randomly create the initial population.
2. Repeat until the maximum number of iterations is reached:
  - (a) Calculate the fitness value of the individuals.
  - (b) Select individuals based on fitness value.
  - (c) Apply genetic operators to obtain new individuals.
  - (d) Replace the old population with the new one, eventually copying the fittest individual(s).
3. Return the best individual.

More in detail, given a set of  $n$  input saliency estimation algorithms  $S = \{S_k\}_{k=1}^n$ , the candidate solutions evolved by GP are built using the set of functional (or non-terminal) symbols  $F$  and the set of terminal symbols  $T = S$ . The functional symbols correspond to operations performed on the inputs. The list of operations given in Table 3.5 are incorporated into the GP framework, along with their functional symbols. They can be grouped on the basis of the support on which they operate: the first group operates in the 2D spatial neighborhood of the pixels belonging to the same saliency map; the second group operates on stacks of pixels across different saliency maps; the third one operates on individual pixels without considering any neighborhood.

The fitness function is here defined as a weighted average of Mean Absolute Error (MAE) and  $F$ -measure ( $F_\beta$ ) as defined in Section 3.2.1, following the procedure outlined

in [98]. MAE is computed as the average of all the individual MAEs on the training set images.  $F_\beta$  is computed as follows: i) for each possible threshold, Precision and Recall are computed for every training image; ii) Precision and Recall are averaged over all the training images; iii) for every threshold,  $F_\beta$  is computed with  $\beta = \sqrt{0.3}$ ; iv) the maximum  $F_\beta$  across all the thresholds is returned.

Mathematically, the fitness function is defined as:

$$f = w_1 \cdot \sqrt{F_\beta} + w_2 \cdot \text{MAE} \quad (3.5)$$

with  $[w_1, w_2] = [1, -0.01]$ . The weights are chosen with opposite signs since  $F_\beta$  should be maximized while MAE minimized: in this way fitter individuals correspond to larger fitness values. The magnitudes of the weights have been empirically chosen on the basis of the difficulty to optimize the corresponding measures, thus with  $F_\beta$  driving the optimization process.

#### 3.3.2 Back-propagation optimization

The fusion tree generated from the genetic programming step involves different types of operations. The intention is to further refine some of these operations by exploiting back-propagation optimization. In order to do so, it is necessary to build the corresponding backbone CNN architecture, by representing all tree nodes with differentiable functions, for the gradients to correctly flow up to the input saliency maps. The resulting neural network can also be augmented with further operations, defining an extended CNN. Both backbone CNN and extended CNN are designed as a generalization of the behavior defined by GP, i.e. in their initialization state they produce the same output. Via back-propagation, then, the CNNs can be optimized to generate more accurate solutions.

#### Conversion from fusion tree into CNN

Details of the mapping between operations in the fusion tree and the corresponding CNN are presented in the following.

- **MaxStack and MinStack:**

MaxStack is implemented as 3-dimensional max-pooling with kernel of spatial size  $1 \times 1$ , and depth  $D$  equal to the number of input channels.

MinStack is implemented through 3D min-pooling, i.e. by changing the sign to the input and output of MaxStack as just defined.

- **Dilation and Erosion:**

Greyscale dilation with a squared structural element corresponds to 2-dimensional max-pooling. The kernel has spatial size  $5 \times 5$  as in the genetic programming setup. Erosion is obtained through 2D min-pooling, i.e. by changing the sign to the input and output of dilation.

- **AverageStack:**

Stacked-channel average is implemented as a convolutional layer with one kernel of size  $1 \times 1$ , depth  $I$  equal to the number of input channels determined by GP, and bias initialized at 0. For the extended CNN, two different behaviors are imposed, depending on the location of the node:

- Intermediate node:

Convolutional layer with one kernel of spatial size  $N \times N$ , and depth  $D$ . The best value for  $N$  is defined experimentally.

- Input node:

Convolutional layer with a bank of  $M$  filters with size  $N \times N$ , and depth  $S$  equal to the number of all input saliency algorithms. The filters are initialized to all zeros, except the middle value of the  $D$  kernel channels selected by GP. The layer is followed by a ReLU non-linearity, and by a compacter module that maps  $M$  channels to 1, which is the expected size for the subsequent operation. The compacter is based on a channel-wise average, minimum, or maximum operation. The effective final operation is experimentally determined.

The weights of this layer are optimized via backpropagation.

- **Quantization:**

Quantization is approximated with a generalized logistic function in its 7-parameter formulation:

$$y = (K - A) \cdot (Q \cdot \exp^{-B \cdot (x - M)} + C)^{(-1/V)} + A \quad (3.6)$$

Hard quantization, used in GP, negatively impacts gradient flow, so it is replaced with a soft quantization implemented as a sigmoid centered in 0.5:

$$\begin{array}{llll} A = 0.0 & B = 4.0 & V = 1.0 & C = 1.0 \\ K = 1.0 & Q = 1.0 & M = 0.5 & \end{array}$$

This initial configuration is then optimized through back-propagation.

- **MedianStack:**

Directly implemented as the median operation applied across the depth dimension

### 3.3 Automatic fusion of saliency algorithms

Table 3.6 Comparison of traditional statistics computation (True Positives TP, False Positives FP, False Negatives FN, True Negatives TN) with threshold set to 0.5, and continuous statistics computation. Highlighted in boldface the different behavior between the two approaches.

| PR  | GT | TP       | FP       | FN       | TN       | TP'        | FP'        | FN'        | TN'        |
|-----|----|----------|----------|----------|----------|------------|------------|------------|------------|
| 0.0 | 0  | 0        | 0        | 0        | 1        | 0.0        | 0.0        | 0.0        | 1.0        |
| 0.4 | 0  | 0        | <b>0</b> | 0        | <b>1</b> | 0.0        | <b>0.4</b> | 0.0        | <b>0.6</b> |
| 0.6 | 0  | 0        | <b>1</b> | 0        | <b>0</b> | 0.0        | <b>0.6</b> | 0.0        | <b>0.4</b> |
| 1.0 | 0  | 0        | 1        | 0        | 0        | 0.0        | 1.0        | 0.0        | 0.0        |
| 0.0 | 1  | 0        | 0        | 1        | 0        | 0.0        | 0.0        | 1.0        | 0.0        |
| 0.4 | 1  | <b>0</b> | 0        | <b>1</b> | 0        | <b>0.4</b> | 0.0        | <b>0.6</b> | 0.0        |
| 0.6 | 1  | <b>1</b> | 0        | <b>0</b> | 0        | <b>0.6</b> | 0.0        | <b>0.4</b> | 0.0        |
| 1.0 | 1  | 1        | 0        | 0        | 0        | 1.0        | 0.0        | 0.0        | 0.0        |

(i.e. each image coordinate is determined as the median of the corresponding coordinates from input channels).

- **Median:**

The median filter is implemented in a two-step processing.

The input is unfolded once per each spatial dimension, i.e. sliding windows are explicitly replicated using unitary stride and kernel size  $5 \times 5$ .

The copies are unrolled along an extra dimension, where the median operation is applied, effectively producing one value per sliding window location.

At the end of the tree, a final non-linearity is introduced using a logistic function in the form of Equation 3.6, followed by a clamping of the values between 0 and 1. In this case, the logistic is initialized as an approximation of the identity function:

$$\begin{aligned}
 A &= -1.557 & B &= 0.666 & V &= 0.354 & C &= 0.817 \\
 K &= 1.165 & Q &= 19.747 & M &= -5.856
 \end{aligned}$$

Clamping replicates the eventual cut-off given by saving the image to file before final application or evaluation. It also allows the logistic curve to create strong distortions inside the  $[0,1]$  range, preventing instead the loss function from penalizing values outside of it.

In the extended CNN, the same “logistic+clamping” block is introduced at each node of the computation (i.e. after each operation).

#### Loss function

The final objective is to optimize both Mean Absolute Error (MAE) and  $F$ -measure ( $F_\beta$ ).  $F_\beta$  is the weighted harmonic mean between precision and recall, therefore relying on a hard thresholding step for computation. MAE is instead evaluated directly on the raw prediction, without requiring any binarization.

While MAE can be employed in its original form as a loss function,  $F_\beta$  is not suitable for back-propagation: according to the procedure defined in [98], in fact, the specific value for threshold is chosen by performing binarization at different levels, and selecting the one that eventually produces the best performance. For this reason, the learning process is here performed with a proposed continuous version of True Positives ( $TP$ ), False Positives ( $FP$ ), and False Negatives ( $FN$ ), in order to avoid both the need for a hard threshold, as well as the process of its selection.  $F_\beta$  can be adapted as a loss function, here called  $F_\beta^{CC}$ , as follows:

1. Let  $TP'$ ,  $FP'$ , and  $FN'$  be the continuous variants of  $TP$ ,  $FP$  and  $FN$ :

$$TP' = PR \cdot GT \quad (3.7)$$

$$FP' = PR \cdot (1 - GT) \quad (3.8)$$

$$FN' = (1 - PR) \cdot GT \quad (3.9)$$

Where  $PR$  is the saliency prediction on each pixel, and  $GT$  the corresponding ground truth. The effect of this formulation can be observed in Table 3.6.

2. The above measures are summed over all pixels  $p$  in each image  $i$ , and the continuous variants of precision and recall are computed as:

$$\begin{aligned} Precision'_i &= \frac{\sum_p TP'_{p,i}}{\sum_p TP'_{p,i} + \sum_p FP'_{p,i}} = \\ &= \frac{\sum_p (PR_{p,i} \cdot GT_{p,i})}{\sum_p (PR_{p,i} \cdot GT_{p,i}) + \sum_p [PR_{p,i} \cdot (1 - GT_{p,i})]} = \\ &= \frac{\sum_p PR_{p,i} \cdot GT_{p,i}}{\sum_p PR_{p,i}} \end{aligned} \quad (3.10)$$

$$\begin{aligned}
 Recall'_i &= \frac{\sum_p TP'_{p,i}}{\sum_p TP'_{p,i} + \sum_p FN'_{p,i}} = \\
 &= \frac{\sum_p (PR_{p,i} \cdot GT_{p,i})}{\sum_p (PR_{p,i} \cdot GT_{p,i}) + \sum_p [(1 - PR_{p,i}) \cdot GT_{p,i}]} = \\
 &= \frac{\sum_p PR_{p,i} \cdot GT_{p,i}}{\sum_p GT_{p,i}}
 \end{aligned} \tag{3.11}$$

3. The continuous precision and recall values are averaged among all images in the set  $I$ , as done in the final evaluation procedure [98]:

$$Precision' = \frac{1}{|I|} \sum_i Precision'_i \tag{3.12}$$

$$Recall' = \frac{1}{|I|} \sum_i Recall'_i \tag{3.13}$$

4. A continuous variant of  $F_\beta$  is computed from these values using fixed  $\beta = \sqrt{0.3}$ :

$$F_\beta^C = \frac{(1 + \beta^2) Precision' \cdot Recall'}{\beta^2 Precision' + Recall'} \tag{3.14}$$

5.  $F_\beta^C$  is complemented in order to effectively obtain a measure related to error:

$$F_\beta^{CC} = 1 - F_\beta^C \tag{3.15}$$

MAE and  $F_\beta^{CC}$  (complemented continuous  $F_\beta$ ) are then combined into a unique loss by means of a weighted average. Since it is not desirable, in principle, to have one measure overweighting the other, the weight is determined by bringing the two losses to the same value at the beginning of the back-propagation optimization process, i.e. in the sub-optimal configuration proposed by the genetic optimization.

## 3.4 Experiments

In this section, the experimental setup is first described by introducing the input saliency estimation algorithms, the datasets that have been adopted at different phases of the optimization, and the evaluation metrics. The following experiments are then presented: different fusion trees are selected from the genetic programming phase, the corresponding CNNs are generated, and finally evaluated on various datasets for a comparison with the input algorithms.

### 3.4.1 Setup

Most of the data-driven methods behind the exploited input saliency maps were originally trained on the training set of the MSRAB dataset [130] or its variants, as reported in [98]. The process of optimizing the best fusion trees, instead, has been here performed on a subset of the validation set of MSRAB. The prediction quality of the input methods, in fact, is expected to be slightly lower on “new” images, so it is desirable to emulate, during the optimization process, the actual input distribution that will be obtained at inference time. Precomputed saliency maps for the input algorithms are available, in some cases, only on the MSRA10K dataset [44], which shares 3756 input images with MSRAB. The MSRAB Validation Subset and MSRAB Test Subset have been therefore defined, based on the intersection between the original splits of MSRAB and the entire MSRA10K dataset, in order to establish a common dataset among all input algorithms. The Validation Subset has been used for optimization, and the Test Subset for performance comparison against the input algorithms.

Experiments have been conducted on handcrafted and deep learning input saliency algorithms as shown in Table 3.7. The FCN-based solution presented in the preliminary investigation from Section 3.2 is here referred to as MFCN. Table 3.7 also reports the availability of already computed saliency maps for different algorithms on different datasets.

Special attention has been given to deep learning methods, which are further tested on additional datasets according to [98], namely: DUTOMRON [205], ECSSD [204], HKU-IS [120], PASCALS [125] and SOD [140]. Whenever precomputed saliency maps for a given deep learning method are not available for any one of the involved datasets, they have been recomputed locally (on all datasets) with official code implementations. This has been done to avoid exposing the model to different behaviors across tests.

All solutions have been evaluated according to MAE and  $F_\beta$  as defined in Section 3.2.1, using the evaluation code provided with [98].

### 3.4.2 Optimization results

Two different experiments have been conducted. In the first one, genetic programming is performed only on saliency estimation algorithms that are based on hand-crafted features. In the second one input maps coming from deep learning based saliency estimation algorithms are instead considered. The aim of the first experiment is to assess how much the proposed method can improve over individual methods, and how it compares with

### 3.4 Experiments

Table 3.7 Input saliency algorithms on the various datasets that have been used at different phases of optimization and evaluation. The symbols denote, respectively,  $\circ$ : unavailability of official saliency maps,  $\bullet$ : only partial availability,  $\bullet$ : full availability. The last column indicates whether saliency maps have been recomputed with official code (on all datasets) on grounds of the observed availability.

| Datasets           |            | M [130]   | D [205]   | E [204]   | H [120]   | P [125]   | S [140]   | Computed locally |
|--------------------|------------|-----------|-----------|-----------|-----------|-----------|-----------|------------------|
| Methods            |            |           |           |           |           |           |           |                  |
| Handcrafted (HC)   | DRFI [193] | $\bullet$ | $\bullet$ | $\bullet$ | $\circ$   | $\bullet$ | $\circ$   |                  |
|                    | DSR [123]  | $\bullet$ | $\bullet$ | $\bullet$ | $\circ$   | $\bullet$ | $\circ$   |                  |
|                    | EQC [10]   | $\circ$   | $\circ$   | $\circ$   | $\circ$   | $\circ$   | $\circ$   | ✓                |
|                    | GMR [205]  | $\bullet$ | $\bullet$ | $\bullet$ | $\circ$   | $\bullet$ | $\circ$   |                  |
|                    | HDCT [109] | $\circ$   | $\circ$   | $\circ$   | $\circ$   | $\circ$   | $\circ$   | ✓                |
|                    | MB+ [211]  | $\circ$   | $\circ$   | $\circ$   | $\circ$   | $\circ$   | $\circ$   | ✓                |
|                    | MC [105]   | $\bullet$ | $\bullet$ | $\bullet$ | $\circ$   | $\bullet$ | $\circ$   |                  |
|                    | RBD [220]  | $\circ$   | $\circ$   | $\circ$   | $\circ$   | $\circ$   | $\circ$   | ✓                |
|                    | RC [44]    | $\bullet$ | $\bullet$ | $\bullet$ | $\circ$   | $\bullet$ | $\circ$   |                  |
|                    | ST [132]   | $\circ$   | $\circ$   | $\circ$   | $\circ$   | $\circ$   | $\circ$   | ✓                |
| Deep learning (DL) | DCL [121]  | $\bullet$ | $\bullet$ | $\bullet$ | $\bullet$ | $\circ$   | $\circ$   | ✓                |
|                    | DHS [128]  | $\circ$   | $\circ$   | $\bullet$ | $\circ$   | $\bullet$ | $\circ$   | ✓                |
|                    | DS [124]   | $\bullet$ | $\bullet$ | $\bullet$ | $\circ$   | $\bullet$ | $\bullet$ | ✓                |
|                    | DRCN [129] | $\circ$   | $\circ$   | $\circ$   | $\circ$   | $\circ$   | $\circ$   | ✓                |
|                    | DSS [98]   | $\bullet$ | $\bullet$ | $\bullet$ | $\bullet$ | $\bullet$ | $\bullet$ | ✓                |
|                    | ELD [116]  | $\circ$   | $\bullet$ | $\bullet$ | $\circ$   | $\bullet$ | $\circ$   | ✓                |
|                    | MDF [120]  | $\bullet$ | $\bullet$ | $\bullet$ | $\bullet$ | $\bullet$ | $\circ$   | ✓                |
|                    | MFCN [18]  | $\bullet$ | $\bullet$ | $\bullet$ | $\bullet$ | $\bullet$ | $\bullet$ |                  |
|                    | RFCN [195] | $\circ$   | $\circ$   | $\circ$   | $\circ$   | $\circ$   | $\circ$   | ✓                |
|                    | SC [216]   | $\circ$   | $\circ$   | $\circ$   | $\circ$   | $\circ$   | $\circ$   | ✓                |

deep learning based algorithms. The second experiment aims to assess if the proposed method is able to improve also the results of state of the art algorithms.

For the first experiment, the fittest individual at the end of the GP optimization is here analyzed. The fusion tree found, named HC-f , is reported in Figure 3.7(a). From the analysis of the operation tree it can be seen that HC-f selected only five out of the ten input saliency estimation algorithms available. The results obtained by the HC-f fusion tree, HC-f backbone CNN and HC-f extended CNN on the MSRAB-Validation set are reported in Table 3.8. From the results it is possible to see how the backbone CNN version of HC-f is able to reduce MAE by almost 57% with a negligible reduction of  $F_\beta$ . The extended CNN further reduces MAE of the backbone CNN by almost 5% at the same time increasing  $F_\beta$ .

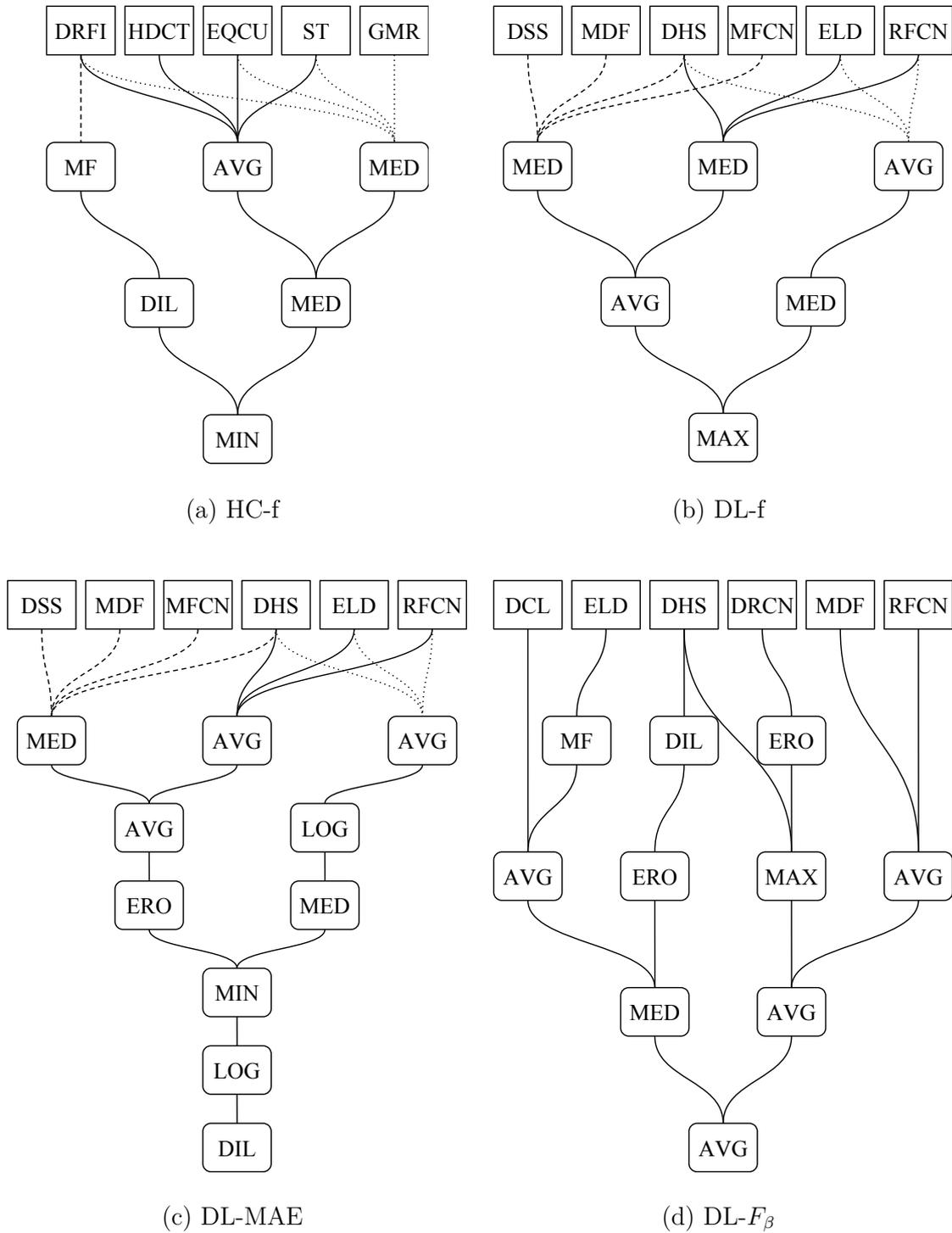


Fig. 3.7 Fusion trees generated through Genetic Programming. Tree (a) is based on handcrafted input saliency algorithms, while trees (b) (c) and (d) are based on deep learning saliency algorithms.

Table 3.8 Results for fusion trees at different levels of optimization on the MSRAB Validation Subset. Lower MAE is better, higher  $F_\beta$  is better.

|               | Optimization | MAE    | $F_\beta$ |
|---------------|--------------|--------|-----------|
| HC-f          | Fusion tree  | 0.1143 | 0.9040    |
|               | Backbone CNN | 0.0494 | 0.9039    |
|               | Extended CNN | 0.0471 | 0.9074    |
| DL-f          | Fusion tree  | 0.0412 | 0.9457    |
|               | Backbone CNN | 0.0259 | 0.9454    |
|               | Extended CNN | 0.0260 | 0.9439    |
| DL-MAE        | Fusion tree  | 0.0284 | 0.9447    |
|               | Backbone CNN | 0.0258 | 0.9457    |
|               | Extended CNN | 0.0253 | 0.9446    |
| DL- $F_\beta$ | Fusion tree  | 0.0903 | 0.9464    |
|               | Backbone CNN | 0.0267 | 0.9467    |
|               | Extended CNN | 0.0252 | 0.9448    |

The second experiment is based on the analysis of three individuals within the population at the last iteration of the GP optimization: the fittest individual (named DL- $f$ ), the individual with the lowest MAE (named DL-MAE), and the individual with the maximum  $F_\beta$  (named DL- $F_\beta$ ). The corresponding fusion trees are depicted in Figure 3.7(b) to (c). From the analysis of the corresponding trees it can be seen that all the three solutions selected only six out of the ten input saliency estimation algorithms available. Focusing on DL-MAE, which has the largest variety of operations used, Figure 3.8 reports the corresponding fusion tree, its conversion into the backbone CNN and its extended CNN. The results obtained by the fusion tree, backbone CNN and extended CNN of the DL- $f$ , DL-MAE and DL- $F_\beta$  on the MSRAB Validation Subset are reported in Table 3.8. From the results it can be noticed that the backbone CNNs produce further improvements of MAE with respect to the corresponding fusion trees, with the most relevant being DL- $F_\beta$ , which sees a drop in error by almost 72%.  $F_\beta$  presents a small average improvement across all backbone CNNs. On the contrary, the extended CNNs tend to produce overall slightly lower values for  $F_\beta$ . This, however, is here considered to be an acceptable compromise to reach a further level of reduction for MAE.

Table 3.9 reports the results of the four extended CNNs on the MSRAB Test Subset, in a comparison with the input saliency estimation algorithms. HC-f outperforms all other handcrafted methods, reducing MAE on the second best (MB+) by 50% and

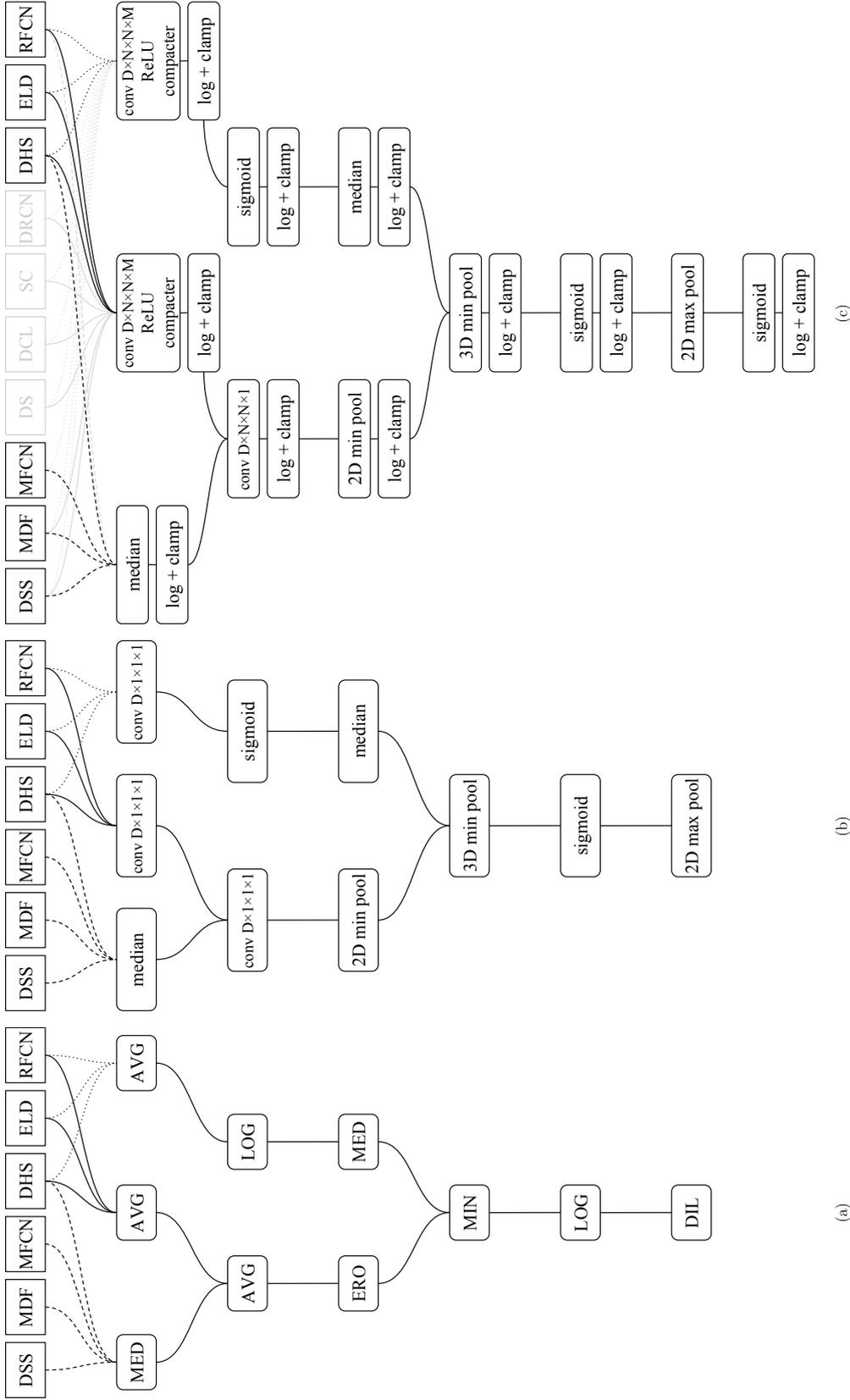


Fig. 3.8 Different implementation stages of the automatically designed DL-MAE : (a) fusion tree, (b) backbone CNN, and (c) extended CNN. Convolutional layers are described with  $D \times N \times N \times M$ , where  $D$  is the filter depth (number of input channels),  $N$  denotes the filter spatial size, and  $M$  is the cardinality of filter bank (number of output channels).

Table 3.9 Comparison of the optimized fusion trees with input saliency algorithms on the MSRAB Test Subset. Lower MAE is better, higher  $F_\beta$  is better.

|      | MAE           | $F_\beta$     |               | MAE           | $F_\beta$     |
|------|---------------|---------------|---------------|---------------|---------------|
| DRFI | 0.1201        | 0.8655        | DSS           | 0.0461        | 0.9226        |
| DSR  | 0.1227        | 0.8237        | MFCN          | 0.0738        | 0.9138        |
| EQC  | 0.1150        | 0.8612        | DCL           | 0.0590        | 0.9133        |
| GMR  | 0.1251        | 0.8375        | DHS           | 0.0326        | 0.9408        |
| HDCT | 0.1477        | 0.8340        | DS            | 0.0665        | 0.9069        |
| MB+  | 0.1095        | 0.8390        | DRCN          | 0.2065        | 0.7326        |
| MC   | 0.1435        | 0.8404        | ELD           | 0.0400        | 0.9229        |
| RBD  | 0.1120        | 0.8367        | MDF           | 0.0793        | 0.8765        |
| RC   | 0.1378        | 0.8354        | RFCN          | 0.0605        | 0.9365        |
| ST   | 0.1251        | 0.8583        | SC            | 0.0572        | 0.8871        |
| HC-f | <b>0.0544</b> | <b>0.8971</b> | DL-MAE        | <b>0.0250</b> | <b>0.9467</b> |
|      |               |               | DL- $F_\beta$ | 0.0253        | 0.9462        |
|      |               |               | DL-f          | 0.0253        | <b>0.9467</b> |

(a)
(b)

improving  $F_\beta$  of DRFI by 4%. The same CNN also results in better performance than 7 out of 10 deep learning methods, according to MAE, and better than three of them according to  $F_\beta$ . This suggests to what extent it is possible to reach deep-learning-level performance by proper fusion of handcrafted solutions.

The three deep learning fusion trees once again exhibit similar performance with respect to each other, and produce the overall best results in a comparison with the input saliency methods. DL-MAE in particular outperforms the second best deep learning method (DHS) in reducing MAE by 23%, and improving  $F_\beta$  by less than 1%. This is a further manifestation of the difficulties encountered in significantly improving the  $F_\beta$  measure.

### 3.4.3 Extension to other datasets

With the intent of focusing on the most effective solutions, the deep learning input algorithms and the resulting fusion CNNs have been also evaluated on other standard datasets for saliency estimation, namely DUTOMRON [205], ECSSD [204], HKU-IS [120], PASCALS [125], and SOD [140]. Each of these datasets has been annotated with potentially different criteria, as noted in the introduction of this chapter, so they constitute

### 3.4 Experiments

Table 3.10 Comparison of deep learning fusion trees and input saliency algorithms on various datasets. Lower MAE is better, higher  $F_\beta$  is better.

|               | DUTOMRON     |              | ECSSD        |              | HKU-IS       |              | PASCALS      |              | SOD          |              |
|---------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|
|               | MAE          | $F_\beta$    |
| DSS           | 0.074        | 0.761        | 0.065        | 0.906        | 0.050        | 0.900        | 0.102        | 0.823        | 0.126        | 0.836        |
| MFCN          | 0.163        | 0.702        | 0.118        | 0.865        | 0.115        | 0.852        | 0.161        | 0.783        | 0.182        | 0.769        |
| DCL           | 0.094        | 0.733        | 0.080        | 0.896        | 0.063        | 0.893        | 0.115        | 0.807        | 0.131        | 0.833        |
| DHS           | <b>0.027</b> | <b>0.909</b> | 0.062        | 0.905        | 0.052        | 0.892        | 0.092        | 0.828        | 0.128        | 0.825        |
| DS            | 0.084        | 0.774        | 0.080        | 0.900        | 0.079        | 0.866        | 0.109        | 0.830        | 0.127        | 0.832        |
| DRCN          | 0.412        | 0.417        | 0.343        | 0.625        | 0.337        | 0.578        | 0.329        | 0.595        | 0.342        | 0.619        |
| ELD           | 0.092        | 0.738        | 0.082        | 0.865        | 0.072        | 0.843        | 0.122        | 0.774        | 0.155        | 0.765        |
| MDF           | 0.098        | 0.697        | 0.114        | 0.831        | 0.095        | 0.824        | 0.145        | 0.764        | 0.164        | 0.786        |
| RFCN          | 0.094        | 0.747        | 0.097        | 0.898        | 0.078        | 0.895        | 0.118        | 0.829        | 0.161        | 0.807        |
| SC            | 0.098        | 0.676        | 0.106        | 0.816        | 0.098        | 0.772        | 0.148        | 0.712        | 0.182        | 0.704        |
| DL-MAE        | 0.042        | 0.844        | 0.052        | 0.920        | 0.039        | 0.913        | 0.081        | 0.849        | 0.128        | 0.836        |
| DL- $F_\beta$ | 0.041        | 0.850        | <b>0.052</b> | <b>0.921</b> | <b>0.038</b> | <b>0.915</b> | <b>0.079</b> | <b>0.855</b> | <b>0.126</b> | <b>0.842</b> |
| DL-f          | 0.043        | 0.844        | 0.055        | 0.919        | 0.041        | 0.911        | 0.083        | 0.851        | 0.132        | 0.833        |

an interesting benchmark to assess the generalization capabilities of the proposed fusion strategy. Results are reported in Table 3.10.

Although the three optimized CNNs exhibit comparable performance, there is a subtle yet consistent advantage of DL- $F_\beta$  over DL-MAE, differently from what was observed in the previous experiments. DL-MAE has therefore slightly overfitted on the annotation choices that are specific to the MSRAB dataset, while DL- $F_\beta$  produced a better generalization. Overall, the CNNs obtained through the optimization process appear to improve on the input algorithms on all datasets, with the exception of DUTOMRON, where DHS stands out from all other reported performance values. However, by further inspection, according to [128] and [98] DHS was trained on both the MSRA10K dataset and the DUTOMRON dataset itself, thus explaining the observed outlying performance.

Figure 3.9 shows the effect of DL-MAE on different sample images, along with the exploited input saliency maps coming from the corresponding saliency estimation algorithms.

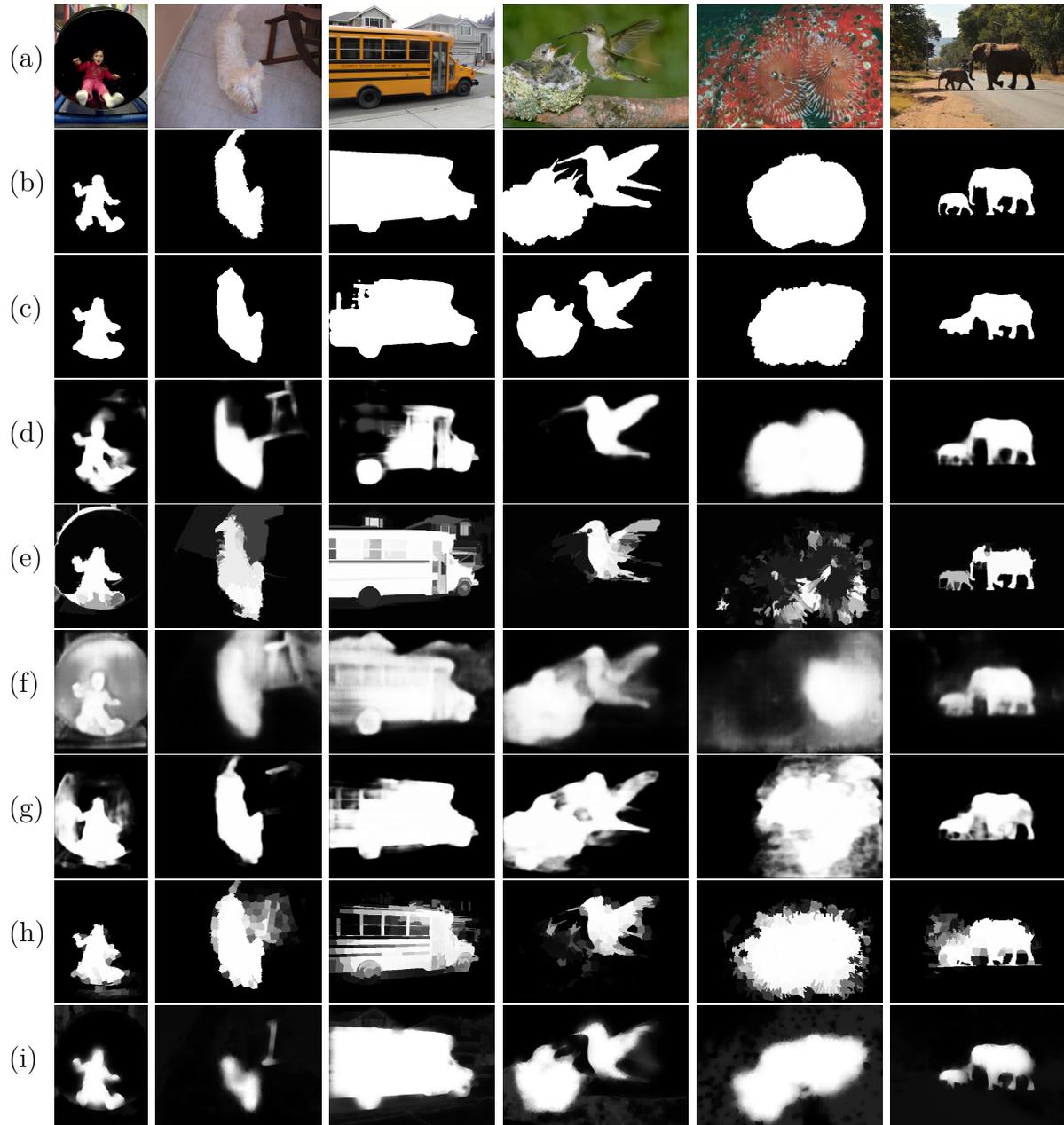


Fig. 3.9 Visual results of the extended CNN from DL-MAE . Rows (a) and (b) show, respectively, the starting image and the corresponding ground truth annotation. (c) is the output of DL-MAE . Rows (d) to (i) are the input saliency maps used by DL-MAE , namely: (d) DSS, (e) MDF, (f) MFCN, (g) DHS, (h) ELD, (i) RFCN.

# Chapter 4

## Semantic segmentation

*“La Golf non c’è, poi mi pare di trovarla [...] e sotto ad un riflettore la riconosco. Tutta brillante nel suo grigiore metallizzato.”*

Offlaga Disco Pax – Dove ho messo la Golf?

Semantic segmentation integrates traditional image segmentation (which partitions the input image into subregions without a regular shape) with the categorical classification of each generated subregion. In the adopted pipeline for image understanding, semantic segmentation falls in-between the steps of object proposal and classification. The most recent and successful approaches to the problem [134, 39, 40, 42], in fact, do not explicitly separate the two elements into disjoint processings, but rely on an end-to-end solution that directly produces the expected output, i.e. a pixel-level classification of the input image.

The first successful approach to semantic segmentation of recent years is the Fully Convolutional Network (FCN) [134], already covered in Chapter 3. This solution exploits the intermediate activations of a deep model previously trained on object recognition, in order to provide an analysis of the input image at different levels of abstraction. The FCN model was originally developed, and tested, on the 20 object classes from the PASCAL Visual Object Challenge [66]. The purpose of this chapter is instead to design a CNN architecture that can provide semantic segmentation and a global scene understanding in outdoor street environments. The automotive field, in fact, has seen a strong expansion in recent years, where a crucial role is played by perception systems for autonomous vehicles and for assisted driving. The focus is therefore on semantic segmentation of street images for car-specific applications, where a model is continuously run on vehicles in order to quickly make decisions in reaction to environmental events. For this reason,

every architectural choice emerges from a compromise between precision and processing speed, with the intent of building an efficient architecture based on a lightweight decoder.

The designed network architecture, which relies on multiresolution analysis, achieves the desired balancing between speed and accuracy. For its development, an incremental approach has been used, highlighting the advantages and disadvantages of each choice.

## 4.1 Related works

The great majority of current image segmentation architectures are based on the encoder-decoder structure [134] to exploit a large receptive field and, at the same time, to produce high-resolution predictions. Architectures that use dilated convolutions [209, 214, 42] to expand the receptive field, also adopt some form of downsampling to keep the computational effort low. Semantic maps are generally produced at 1/8 or 1/16 of the final resolution, and are subsequently upsampled using either nearest or bilinear interpolation.

A good compromise between processing speed and accuracy is difficult to achieve, and it heavily depends on the final application. For this reason, existing works can typically be categorized into two classes: accuracy-oriented architectures, and efficiency-oriented architectures.

### 4.1.1 Accuracy-oriented architectures

The first work that successfully used CNNs for semantic segmentation is FCN [134]. Authors used a pre-trained encoder in conjunction with a simple decoding module, applying skip connections from the lower-level layers to process high-resolution activation maps. DeepLab [39] included Dilated Convolutions [208] to increase the context awareness (i.e. receptive field) of inner layers, while keeping a low number of parameters. Several methods adopted the Residual Network architecture [91] as encoder (such as DeepLabv2 [40], Resnet38 [200], FRRN [156]), further improving performance on the task of semantic segmentation. DeepLabv3 [42] and PSPNet [214] introduced the concept of context layers to further expand the theoretical receptive field of internal activations. These methods achieve high accuracy on different benchmarks but at the expense of high computational costs.

### 4.1.2 Efficiency-oriented architectures

ENet [152] was developed as a high-speed architecture, drastically increasing the efficiency of the model, but sacrificing accuracy. ERFNet [169] adopted a very simple encoder-decoder structure inspired by ENet. ERFNet authors designed the network structure on Residual Factorial convolutions that efficiently process the input signal with dedicated filters for each spatial dimension. SegNet [11] exploits high-resolution information by saving max-pooling indices at the encoding stage, and subsequently using them during the decoding phase. The design of ICNet [213] is based on a three-branch architecture with deep training supervision. The authors also experimented with a form of model compression to further accelerate the network.

## 4.2 Dataset and evaluation metrics

Experiments were carried out on the Cityscapes [55] dataset: a set of urban street images annotated with pixel-wise semantic information. It is composed of 5000 high-resolution images ( $2048 \times 1024$ ) out of which 2975, 500 and 1525 images belong respectively to train, validation and test subsets. Annotations include 30 different classes of objects, although only 19 are typically used for training and evaluation:

- road
- sidewalk
- building
- wall
- fence
- pole
- traffic light
- traffic sign
- vegetation
- terrain
- sky
- person
- rider
- car
- truck
- bus
- train
- motorcycle
- bicycle
- (background)

The dataset is characterized by a vast diversity of scenes, with images taken from 50 cities during different seasons. Acquisitions are all performed at daytime, with good or medium weather conditions.

Two metrics are used for model validation: *mean of class-wise Intersection over Union (mIoU)* for segmentation quality, and *Frame Per Second (FPS)* for processing speed. mIoU is defined as the mean of IoU computed independently for each class. IoU (also called Jaccard Index) is computed as:

$$IoU = \frac{TP}{TP + FP + FN} \quad (4.1)$$

Where TP, FP and FN are the number of True Positive, False Positive, and False Negative pixels, respectively.

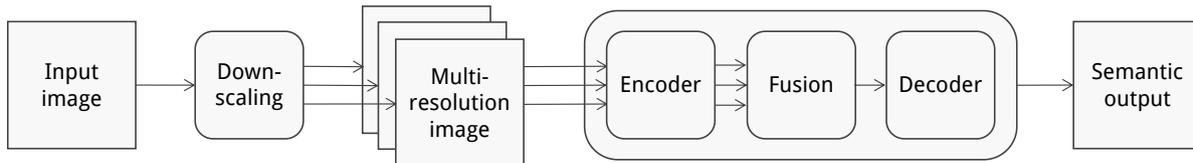


Fig. 4.1 Simplified representation of the proposed architecture for semantic segmentation. The input is processed at different resolutions in order to exploit both details and large receptive fields. The corresponding activation maps are then combined with a fusion module, and fed through a decoder. Further details are provided in Figure 4.2

FPS is defined as the inverse of time necessary for the network to perform one forward pass. All the FPS performance values reported in the following sections are referred to a single NVIDIA Titan Xp GPU.

## 4.3 Network design

The proposed neural network is a multiresolution architecture that jointly exploits high-resolution and large context information to produce an accurate segmentation of the input image. The network relies on an encoder-decoder structure as depicted in Figure 4.1. The encoder is composed by multiple branches with partially shared-weights. The task of these branches is to extract fine and coarse features from the input image subsampled at different resolutions. These multiresolution signals are merged by a *fusion module*, analyzed in Section 4.3.4. The decoder performs a compression in the channels dimension, followed by an upsampling. The output of the network is a class-wise probability distribution for each pixel. The network architecture is presented in details in the following subsections.

### 4.3.1 Evaluation setup

All network configurations have been trained with Stochastic Gradient Descent (SGD) and momentum set at 0.9. Following [209], the base learning rate was set to 0.001 and trained for 250 epochs. A *fixed step* learning rate policy was adopted: the initial value was decreased two times by an order of magnitude, at 100 and 200 epochs. Different starting learning rates were tested, as well as the *poly* learning rate policy from [39], however the baseline configuration provided the best results. Batch size was found to be an important parameter, affecting the final model accuracy: different values were experimented with, resulting in 8 being the best value for this setup. In contrast to what claimed in [41], increasing the batch size negatively affected the overall performance. A

Table 4.1 Impact of downsampling on the system performance. This simple way of speeding-up inference time affects significantly the final accuracy.

|          | Baseline    | Downsampling $\times 4$ |
|----------|-------------|-------------------------|
| mIoU (%) | <b>65.5</b> | 57.5                    |
| FPS      | 6.7         | <b>50.6</b>             |

possible explanation for this phenomenon is that the higher stochasticity introduced by intra-batch dependencies acts as regularizer, improving the final network performance.

### 4.3.2 Input downsampling

Input downsampling is a simple way to speed up the inference process, causing however loss of fine details (e.g. precise borders between classes, or fine textures). To this extent, two simple explorative experiments were set up, in order to investigate a trade-off between system speed and accuracy. A DRN-D-22 model [209] was employed as encoder (with pretraining on Imagenet), and a simple bilinear upsampling was used as decoder. This base model was trained and tested with two different subsampling values. The same model was then trained and evaluated with input images subsampled by factor 4. Table 4.1 shows the mIoU of the baseline model without and without subsampling. Model speed increases from 6.7 FPS to 50.6 but mIoU drops by 8%. This explorative experiment motivated the search for a smarter strategy, based on multiresolution analysis.

### 4.3.3 Multiresolution encoder

A multiresolution network is composed of multiple branches, each processing the input image at a different resolutions. The goal of branches working on lower resolutions is to extract large context features, whereas the goal of branches that work on higher resolutions is to extract more local features that will help to recover fine borders in the decoder. Three different encoder configurations were explored:

1. The first, named *enc24*, consists of two branches that process input images with sub-sampling factors 2 and 4.
2. The second configuration, named *enc24shared*, is similar to the first but weights are shared between the two branches. Results in Table 4.2 show that the network with shared branches achieves higher mIoU levels. Weight sharing between branches, as

| Encoder            | baseline | down $\times$ 4 | enc24 | <b>enc24shared</b> | enc124shared |
|--------------------|----------|-----------------|-------|--------------------|--------------|
| Multiresolution    |          |                 | ✓     | ✓                  | ✓            |
| Shared Parameters  |          |                 |       | ✓                  | ✓            |
| Subsampling factor | 1        | 4               | 2 + 4 | <b>2 + 4</b>       | 1 + 2 + 4    |
| mIoU (%)           | 65.5     | 57.5            | 61.5  | <b>63.0</b>        | 64.2         |
| FPS                | 6.7      | 50.6            | 38.7  | <b>38.7</b>        | 24.9         |

Table 4.2 Performance on Cityscapes validation set and speed (FPS) of different encoder architectures. The first two structures replicate experiments for input downsampling: *baseline* is a full-resolution network, while *down $\times$ 4* is trained and evaluated with down-sampled input. Columns *enc24* and *enc124* means 2 and 3 branches with subsampling factors: (2,4) and (1,2,4) respectively. Column *shared* means that weights are partially shared between branches. In bold the configuration adopted in the final model.

pointed out in [24], induces an implicit form of regularization. This configuration was chosen as the base encoder for the following experiments.

3. In the third configuration, named *enc124shared*, a further branch was added, with the objective of processing the full-resolution image. This brought some performance improvements, but it was not employed in the final model because the whole system would slow down below the real-time threshold of 30FPS.

The different encoder designs in Table 4.2 have been tested together with a fixed decoder architecture. The low-resolution branch is composed of all the layers of a Dilated Residual Network 22 type D (DRN-D-22) [209] with the exception of the last two. The medium-resolution branch has only the first layers of the DRN-D-22 before dilated convolutions. Details are shown in Figure 4.2

#### 4.3.4 Fusion module

The purpose of the fusion module is to join information coming from low-resolution and high-resolution encoder branches. First, the output of the low-resolution branch is upsampled with a differentiable bilinear filter to match the spatial size of the signal coming from the medium-resolution branch. The output of the medium-resolution branch is then expanded from 128 to 512 channels, to match the number of features of the low-resolution branch. Finally, the two multiresolution signals are merged. Table 4.3 reports experimental results of four different designs. Both channel concatenation and addition were evaluated as fusion strategies: concatenation (denoted by “c”) and addition (denoted by “+”). Dimensionality reduction was also evaluated, in order to determine its effect on

### 4.3 Network design

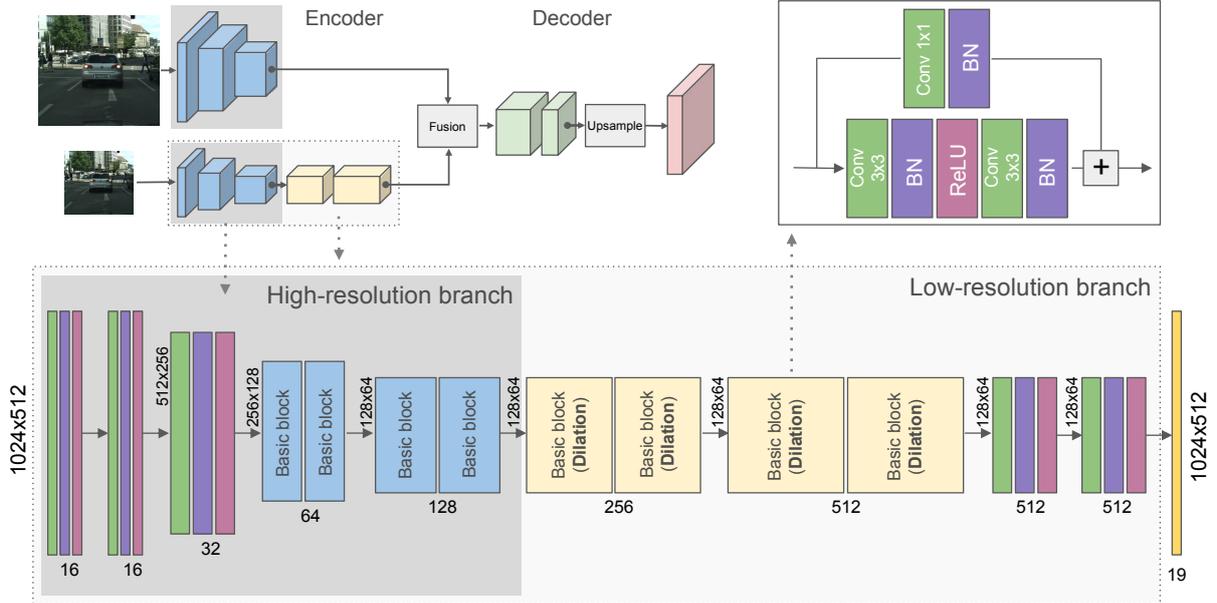


Fig. 4.2 The multiresolution network used for semantic segmentation, shown at incremental levels of detail.

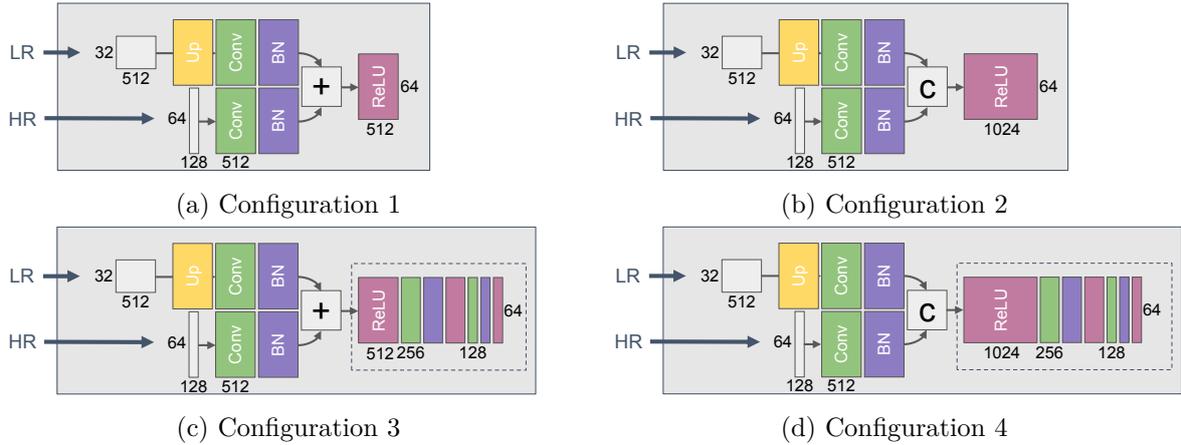


Fig. 4.3 Experimented fusion modules. The four configurations are defined by fusion strategy (addition “+” or concatenation “c”) and presence or absence of postprocessing (denoted by the dotted-line box).

the final performance. This is in opposition to the baseline, where the final classification layer is fed directly with the signal after the fusion operation. Experimental results in Table 4.3 shows that mIoU measure takes advantage of the postprocessing step, whereas the overall speed is only slightly affected. The model benefits from the addition of more non-linearities, and the final upsampling operation is applied to a tensor with a lower channel dimensionality. Figure 4.3 depicts the four different configurations.

| Configuration   | 1    | 2    | 3           | 4    |
|-----------------|------|------|-------------|------|
| Postprocessing  | No   | No   | Yes         | Yes  |
| Fusion strategy | +    | c    | +           | c    |
| mIoU (%)        | 63.0 | 63.5 | <b>65.8</b> | 64.2 |
| FPS             | 38.7 | 37.8 | <b>37.3</b> | 36.4 |

Table 4.3 mIoU on Cityscapes validation set and FPS for different *fusion modules*. Differences are: signal summation or concatenation and presence of a post-processing step. In bold the configuration adopted in the final model.

| Transformation | baseline | color jitter | lighting jitter | <b>random scale</b> |
|----------------|----------|--------------|-----------------|---------------------|
| mIoU (%)       | 65.8     | 62.6         | 64.2            | <b>67.5</b>         |

Table 4.4 mIoU on Cityscapes validation set with different data augmentation techniques used during training. In bold the configuration adopted in the final model.

### 4.3.5 Data augmentation

To make the model more robust to different lighting conditions, several light-related transformations were investigated: this is potentially an important characteristic for the envisioned system, which is expected to work on real environments and outdoor scenes. Two main light transformations were applied: color jitter and lighting Jitter. Color jitter consists of modifying image brightness, saturation and contrast in random-order. Lighting jitter is the same jittering used in [113]. Specifically,  $\sigma = 0.1$  was used as standard deviation to generate random noise. Geometric transformations were also experimented: following [209], images were resized with a random scale factor between 0.5 and 2. Table 4.4 shows the results of adopting these data augmentation techniques. Only *random scale* brought some improvements, which was therefore included in the training procedure.

### 4.3.6 Final architecture

The complete network architecture is reported in Table 4.5. The upper part defines the encoder structure, whereas the bottom part defines the fusion module and the decoder. The network is composed of a total 19 Million parameters, and requires 58.7 GFLOP to perform single inference on a  $512 \times 1024$  image. Most parameters are within the encoder, and in particular in the residual blocks, presented in [209].

A block is a module composed of three layers: Convolution, Batch Normalization and ReLU. In particular, blocks with stride = 2 perform input downsampling. Conv +

### 4.3 Network design

Table 4.5 The proposed network architecture. For each module the output size, number of operations (FLOP) and number of parameters are reported. *Block* is composed by Convolution, BatchNorm and ReLU layers. The word *Dilated* means that convolutions exhibit a dilation term.

| Encoder       |                |      |                  |                   |        |            |
|---------------|----------------|------|------------------|-------------------|--------|------------|
| Branch-1      |                |      | Branch-2         |                   |        | Parameters |
| Type          | Output size    | FLOP | Type             | Output size       | FLOP   |            |
|               |                |      | Subsample        | 256×512×3         | 393,2K | 0          |
| Block         | 512×1024×16    | 1.2G | Block            | 256×512×16        | 308.7M | 2.4K       |
| Block         | 512×1024×16    | 1.2G | Block            | 256×512×16        | 302.4M | 2.3K       |
| Block (s=2)   | 256×512×32     | 609M | Block (stride=2) | 128×256×32        | 302.2M | 4.6K       |
| Res. Block    | 128×256×64     | 3.9G | Residual Block   | 64×128×64         | 973.2M | 131.4K     |
| Res. Block    | 64×128×128     | 3.9G | Residual Block   | 32×64×128         | 973.1M | 524.9K     |
|               |                |      | Dil. Res. Block  | 32×64×256         | 1.9G   | 2.1M       |
|               |                |      | Dil. Res. Block  | 32×64×512         | 7.8G   | 8.4M       |
|               |                |      | Dil. Block       | 32×64×512         | 4.8G   | 2.4M       |
|               |                |      | Block            | 32×64×512         | 4.8G   | 2.4M       |
| Fusion Module |                |      |                  |                   |        |            |
|               |                |      | Upsample         | 64×128×512        | 16.8M  | 0          |
| Conv + BN     | 64×128×512     | 4.8G | Conv + BN        | 64×128×512        | 19.3G  | 3.0M       |
|               | Sum + ReLU     |      |                  | 64×128×512        | 8.4M   | 0          |
|               |                |      |                  | 64×128×256        | 1.1G   | 131.3K     |
|               |                |      |                  | 64×128×128        | 268.5M | 32.9K      |
| Decoder       |                |      |                  |                   |        |            |
|               | Classification |      |                  | 64×128×19         | 19.9M  | 2.5K       |
|               | Upsample       |      |                  | 512×1024×19       | 39.8M  | 0          |
|               |                |      |                  | Total FLOP: 58.7G |        | Total: 19M |

BN is a single block with a Convolution followed by a Batch Normalization layer. The network has two branches in the encoder that share the same weights. For this reason, in Table 4.5 there is only one column with the number of parameters for the two branches. The difference between branches lies in the signal resolution. Branch-2 is fed with a downsampled input, and it is the part of the network with the highest computational burden. Earliest layers of branch-1 are heavier to compute, for this reason branch-1 is shallower by design. The Fusion Module is composed by a first part where signals are pre-processed independently, followed by a second part where signals are jointly processed. The decoder ends with a classification layer (i.e. a  $1\times 1$  filter convolution) followed by bilinear upsampling.

## 4.4 Comparison with other methods

Table 4.6 reports the performance of the proposed architecture along with state of the art methods on Cityscapes test set. Dataset and evaluation metrics correspond to those described in Section 4.2, and the mIoU measure has been computed by the Cityscapes online evaluation server. This table includes only algorithms that declare their running time on the Cityscapes leaderboard (those that do not care about processing time are assumed to be computationally heavy). Most of these methods, e.g. PSPNet, DeepLabv3 [214, 42] achieve very high mIoU levels (DeepLabv3+ is the best-published model to date, reaching 81.2%). However, they typically rely on multiscale testing to increase accuracy, which is particularly time-consuming. The proposed network architecture achieves 68.2% of mIoU on the Cityscapes test set without any postprocessing. FPS in Table 4.6 refer to a single Titan Xp GPU. The proposed network performs even better than some methods like Adelaide [126], Dilation10 [208] etc. that do not take speed into consideration. Only ICNet[213] and ERFNet[169] achieve similar performance in terms of mIoU while being slightly faster. These make use of different strategies to reach high-efficiency predictions which are orthogonal to the model proposed in this chapter. ICNet is based on a similar multiscale architecture, but with a different number of scales and different fusion modules. Authors of ICNet employed auxiliary losses to train the network, while the proposed network can be trained end-to-end with a single cross-entropy loss. In addition to that, to lighten the computational burden of the final model, they compressed the trained model. This technique is orthogonal to the network structure design and could be employed in the exact same way to compress the proposed model. ERFNet model reaches high-efficiency thanks to the use of novel building blocks: Factorized Convolutions. Again, these design choices can be potentially included in the proposed solution to improve efficiency.

## 4.4 Comparison with other methods

Table 4.6 Comparison with state-of-the-art methods on Cityscapes test set.

| Name             | Subsampling | mIoU (%) | FPS   |
|------------------|-------------|----------|-------|
| SegNet [11]      | 4           | 57.0     | 26.4  |
| ENet [152]       | 2           | 58.3     | 121.5 |
| SQ [189]         | no          | 59.8     | 26.4  |
| CRF-RNN [217]    | 2           | 62.5     | 2.2   |
| DeepLab [39]     | 2           | 63.1     | 0.4   |
| FCN-8S [134]     | no          | 65.3     | 4.9   |
| Adelaide [126]   | no          | 66.4     | 0.05  |
| Dilation10 [208] | no          | 67.1     | 0.4   |
| ICNet [213]      | no          | 69.5     | 47.9  |
| ERFNet [169]     | 2           | 69.7     | 62.6  |
| DeepLab v3+ [42] | no          | 82.1     | N/A   |
| This work        | 2           | 68.2     | 43.9  |

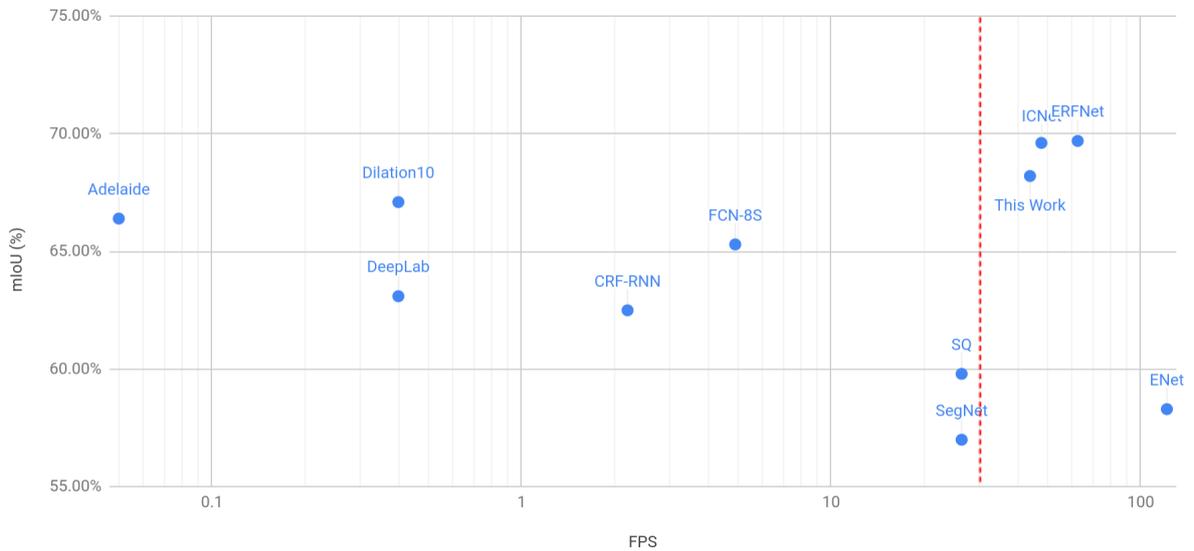


Fig. 4.4 Plot of mIoU vs FPS for comparison of the presented solution with other methods in the state of the art.

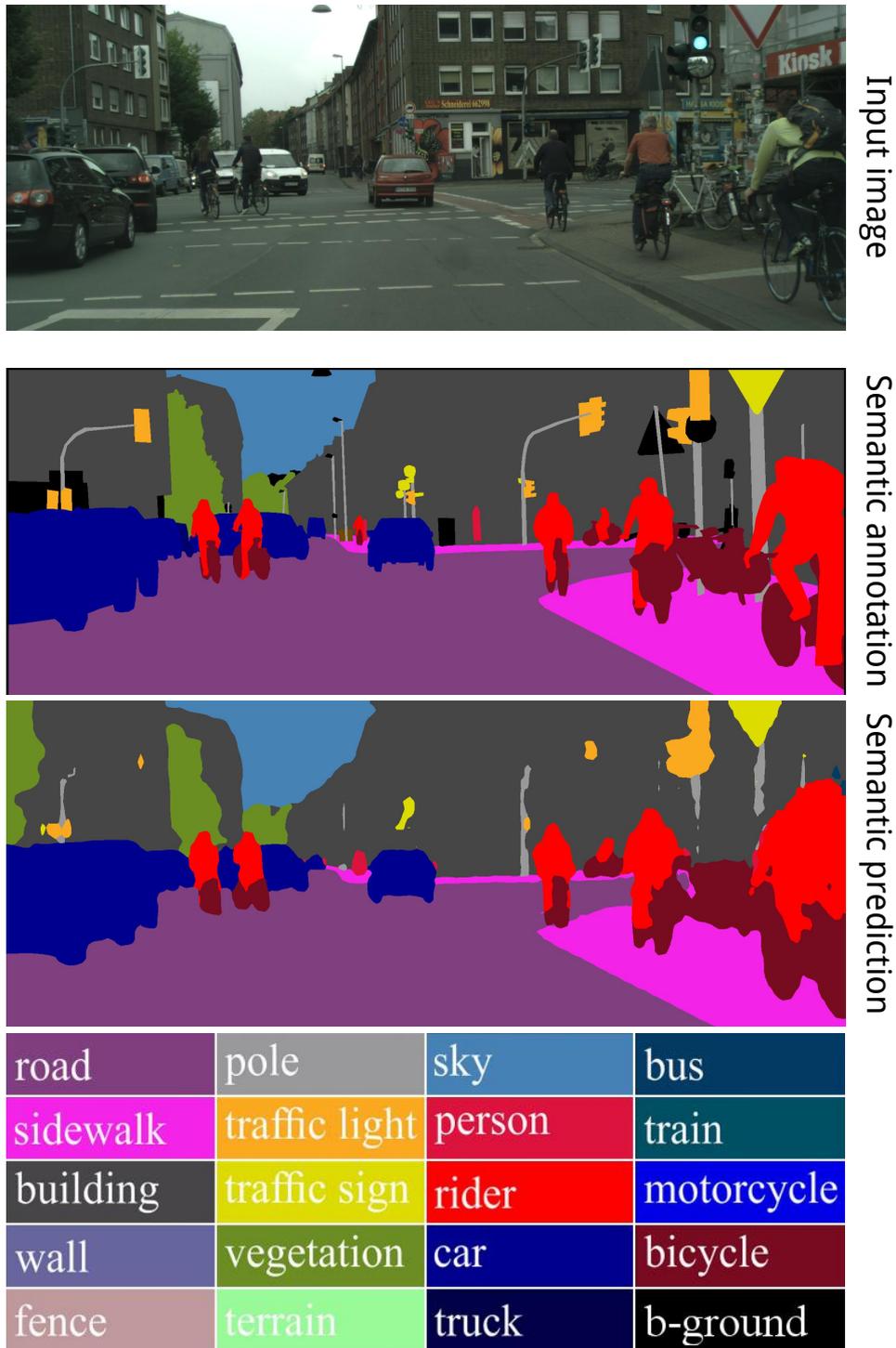


Fig. 4.5 Semantic segmentation results obtained with the proposed network architecture.

**Part II**

**Classification**

# Chapter 5

## Introduction to classification

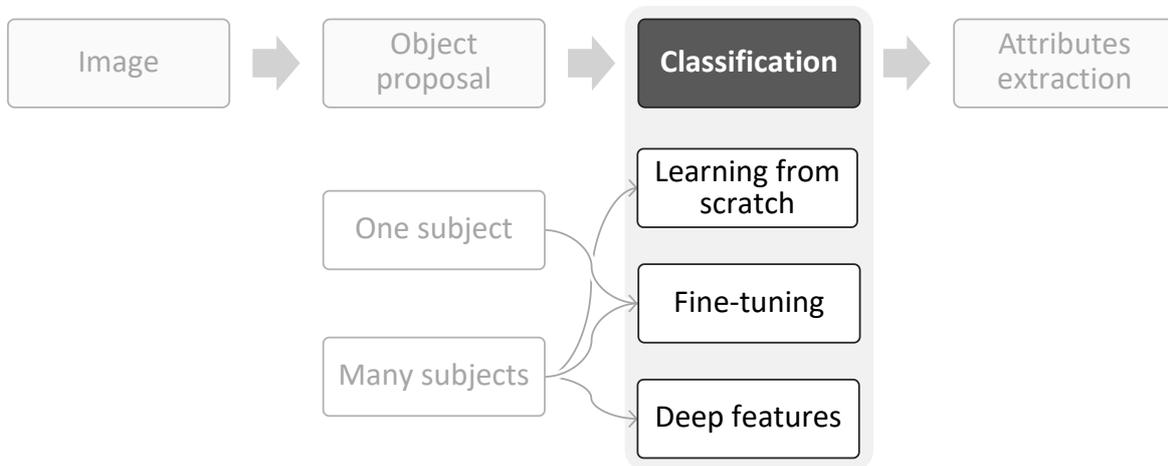


Fig. 5.1 Different approaches for the classification phase.

The second step of the proposed pipeline for automatic description and annotation of complex scenes is the classification of those image parts highlighted by the object proposal phase. This task has been addressed with the use of deep convolutional neural networks, which in later years showed unmatched results in classification tasks [115]. Several architectures have been analyzed and employed at different levels, from the pioneering AlexNet and CIFAR-10 network from [113], moving to the application of the more compact and more efficient ResNet [91]. Eventually, also state-of-the-art solutions like the automatically-optimized networks from NasNet [221] have been exploited, and their generalization capability has been evaluated. As shown in Figure 5.1, neural networks can be employed for classification tasks in, at least, three ways:

- Learning from scratch. When a large amount of labeled data is available, training a deep neural network from scratch is a viable option. In this setup, the learnable

---

weights that characterize the network layers are randomly initialized by resorting to different strategies [83]. The classification loss, typically based on cross-entropy, is used as the entry point for the back-propagation algorithm [172] to incrementally update the weights until convergence to a stable solution is reached.

- **Fine-tuning.** Tightly related to the concept of transfer learning, fine-tuning involves the adoption of a neural network that was pretrained on a different task (typical examples exploit models trained on the ImageNet dataset [173]). For a classification problem, the final fully-connected layer that maps to the original problem cardinality must be replaced with a new untrained layer, mapping to the new, proper, cardinality. The learning process is then actuated with the new training data, and the weights update can be either limited to the new layers, or extended to the whole model. In this case the first layers start training on the new data from a meaningful state, as opposed to random initialization, therefore requiring overall less time for the model to converge.
- **Deep features.** This training modality also consists in exploiting a neural network trained on a different task to process the input image. In particular, activations of layers close to the final output are used as features to describe the input image [82], as they usually constitute compact representations for abstract concepts. Such deep features can be used to train other “traditional” classifiers, e.g. Support Vector Machines [57]. Like fine-tuning, this approach is also especially useful when few training examples are available, as in this situation a full neural network training might lead to an overfitted model.

Availability of training data is of paramount importance in performing supervised training with deep convolutional networks. This is especially the case for learning-from-scratch scenarios, but it covers a significant role for alternative forms of learning as well. The importance of data has been made explicit through several literature works about synthetic and real data augmentation [153, 16]. Synthetic data augmentation reckons on the application of procedural distortions to the training data, with the goal of virtually generating new samples. The distortions can be tailored to the specific characteristics of the problem at hand, as shown in Chapter 6, and generally produce appreciable but limited improvements. Real data augmentation, on the other hand, consists of collecting genuinely new training examples, and of annotating them for the specific task of interest. By nature, this second approach is more expensive to pursue, but it typically has a stronger positive impact on the final recognition performance. This will be also shown experimentally in the following chapters.

---

Two tasks with different characteristics have been selected as uses cases, with the intention of addressing the wider possible spectrum of challenges imposed by real-life applications. The proposed solutions involve all the previously described ways of exploiting neural networks and data augmentation for classification.

The first category of interest regards brand logos [15, 16]. These visual elements are characterized by bright colors and sharp edges, a feature that has been exploited in the following by resorting to the selective search [191] object proposal algorithm for cluttered scenes. In a preliminary study of the problem, classification of logo regions has been first addressed with deep features representation, synthetic forms of data augmentation, and classification via support vector machine. Then, learning from scratch has been made feasible by collecting and annotating an extended training set (real data augmentation), and by defining a more compact neural architecture inspired by the CIFAR-10 challenge [113]. This allowed a significant leap in performance to be achieved, and the individual contribution of each element in the proposed solution has been quantified by careful experimentation.

Classification of vegetables and fruits has been treated as the second case study [34]. As the images in the envisioned domain typically depict one subject (or one class) of interest, object proposal based on saliency estimation has been employed to automatically select an optimal cropping of the input image, prior to the classification itself. For this instance of fine-grained visual classification, transfer learning and synthetic data augmentation have been applied by adapting different neural architectures to the problem at hand, in order to define a robust baseline for subsequent improvement. This preliminary study has then been extended by introducing and evaluating two variants of the fine-tuning process with the objective of exploiting the hierarchical nature of the involved classes.

# Chapter 6

## Classification in cluttered scenes: Logos

*“Sie stellt sich zur Schau für das Konsumprodukt. Und wird von Millionen Augen angeguckt”*

Kraftwerk - Das Model

A logo is the identifying visual symbol of an entity, may it be an association, a brand, or a specific product. As such, a successful logo may be seen every day by millions of people, and a person typically sees thousands of logos and brand-related images every day [70]. There is therefore great interest in automating logo recognition, with varying intentions and applications. The brand owner might be interested in copyright infringement detection [29], as well as automated computation of brand-related statistics on social media [78]. Consumer organizations, on the other hand, are interested in monitoring hidden advertisement from TV shows [4], for instance. Other general applications include intelligent traffic-control systems based on vehicle logos [159], as well as contextual placement with augmented reality [89].

Logos are designed with the objective of catching the eye and being memorable, often relying on bright colors and sharp edges. They are typically printed on flat (or slightly curved) surfaces, effectively constraining the range of visual distortions they might be subject to when captured by an imaging device. All these characteristics can be exploited when building a dedicated system for automatic logo recognition, relying for example on a discriminative color space during the object proposal phase, and developing realistic forms of synthetic data augmentation.

First, a preliminary investigation of the task is presented: describing existing works and showing the efficacy of a basic approach to the problem. This analysis contributes

to the motivations for the proposed solution, which is later defined and whose validity is experimentally shown. Key element in the final solution is the collection and annotation of an extended dataset for logo recognition.

## 6.1 Preliminary investigation

### 6.1.1 Existing works

Logo recognition has been traditionally addressed exploiting keypoint-based descriptors and detectors, which take full advantage of the characteristic sharp-edges of such category [12, 110, 107, 143]. Romberg et al. [168] devised a technique to encode and index the spatial layout of local features present in an image, with the specific intent of performing logo retrieval. In particular, they developed a quantized representation of logo regions based on the composition of basic spatial structures and on local characteristics, such as edges and triangles. Revaud et al. [164] presented a method to rescale the score resulting from logo detection in a noisy context. This was done, once again for retrieval purposes, by learning a burstiness model specific to the provided query logo. Romberg and Lienhart [167] introduced a scalable logo recognition technique based on feature clustering, where each local feature is aggregated along with their spatial neighborhood through a Bag of Words approach. Boia et al. [28, 27] proposed a method to localize and recognize logos through homographic class graphs, and devised a specific solution to handle color variations by using secondary inverse models.

More recently, deep learning has been introduced in tasks related to logo recognition [62, 102]. Eggert et al. [62] applied pre-trained Convolutional Neural Networks and synthetic data augmentation, and experimented with several techniques to solve the problem of limited training examples. A similar approach was also adopted by Iandola et al. [102], where different neural architectures were evaluated. Oliveira et al. [149] included pre-trained neural models into a general recognition pipeline based on quick regions. As a general observation, transfer learning from existing neural networks has been often integrated into logo recognition since its first applications, due to the limited availability of labeled training data.

### 6.1.2 FlickrLogos-32 dataset

For several years, the de-facto standard dataset for logo recognition has been FlickrLogos-32 [167, 168, 164]. Made publicly-available for research purposes, this dataset is composed of 2240 logo images depicting 32 different classes, and 6000 negative (or “background”)



Fig. 6.1 Sample images from the FlickrLogos-32 dataset. For each class, an example was manually selected to show the overall range of variability.

images. For each class, the official split consists of 10 training images, 30 validation images, and 30 test images. Each positive image contains one or more instance of a single logo class, which is depicted as printed on a flat or cylindrical surface, as shown in Figure 6.1. The dataset was originally collected and annotated for logo retrieval in real life applications, and subsequently adopted for logo classification and recognition.

### 6.1.3 Selective search

Selective search exploits a greedy bottom-up grouping of initial image regions, which are generated through a graph-based segmentation of the input image [69]. A similarity measure is first computed between all pairs of neighboring regions. The two most similar regions are merged, and the similarity between the resulting region and existing ones is updated. The cycle is repeated until the whole image is an object proposal. For performance reasons, it is essential that the similarity measure can be recursively computed as a function of previously computed values (i.e. avoiding an explicit recomputation step based on the actual content of the image itself). The whole processing can be performed under a number of color spaces, taking advantage of different invariance properties. This characteristic is resumed in Chapter 6, where the color space that provides the most valuable representation for logo detection is systematically found and exploited.

The similarity among neighbors is computed by analyzing the following characteristics of each region:

- **Color:** A color histogram for each color channel is computed. Neighboring regions are then compared using histogram intersection as similarity measure.
- **Texture:** Gaussian derivatives are computed and aggregated into orientation/channel-specific histograms. The comparison between regions is here also based on histogram intersection.
- **Size:** Smaller regions are given a larger incentive for merging. This has the effect of balancing the grouping, generating object proposals that are well spread across the whole image.
- **Fill:** A pair of regions where one encloses the other, is given higher merging likelihood. Let  $T$  be the tightest bounding box surrounding the two regions, this aspect is captured by measuring the fraction of  $T$  that is not covered by the two individual regions.

The overall similarity is then a sum of the above components. Regions are assigned a hierarchy level  $i$  during this incremental merging process. A region rank is then also computed by multiplying this hierarchy level by a random amount in  $[0, 1]$ . The regions with lower rank are finally removed from the list of generated object proposals.

### 6.1.4 Deep-feature classification

This preliminary investigation was based on an idea first proposed by Girshick et al. [82], which consists in extracting deep-learning-based features from each object proposal using a neural network that was pretrained on a different task, and performing classification on said features using a Support Vector Machine (SVM). The pipeline proposed by Girshick et al. is here extended with query expansion at test time, in order to improve the recognition of instances subject to potential image distortions not seen at training time.

Given an input image, that may contain 0, 1 or multiple logo instances in any size and location, an object proposal phase is first conducted in order to generate a set of image subwindows that are more likely to tightly contain an object of interest (a logo, in this case). For this task, the selective search [191] algorithm has been adopted with the objective of detecting subregions that effectively pop out from the background thanks to color contrast and well-defined edges. This method is also able to propose regions with a diverse range of aspect ratios. Due to perspective transformations of planar logos, in fact, it is not possible to make assumptions on the final appearance of the instances.

Each object proposal is subsequently warped to a common size, set to  $227 \times 227$  pixels, and fed to a neural model that was pretrained on the ILSVRC 2012 task [173] with millions of images annotated for classification into 1000 different classes. For the experiments described in this section, the AlexNet [113] architecture has been exploited: 4096-dimensional features extracted from the activations of the last hidden layer are used as the input to train and apply a multiclass linear Support Vector Machine. The SVM had to discriminate between 33 classes (the original 32 logos, plus a reject class).

In order to take into account the high variability in image transformations the logo instances might be subject to, two strategies have been applied. First, at training time, offline data augmentation is performed according to [153]. This is necessary due to the extremely limited amount of data available, as the FlickrLogos-32 dataset was originally ideated for logo retrieval using SIFT-like features [168]. The following transformations are used in combination, using two possible values for each one: translation, scale, x-axis shear and y-axis shear. Note that by combining the two shear transformations, rotation is virtually introduced in the set of applied transformations, as depicted in Figure 6.2. This operation effectively multiplies the number of training example by a factor of  $\sim 250$ . Secondly, query expansion is performed at test time: whenever an object proposal is selected, three more variations are generated by rotating the region by  $90^\circ$ ,  $180^\circ$ , and  $270^\circ$ . Each alternative view is then fed to the SVM for classification, and the object proposal is assigned the class with a higher confidence score.

### 6.1.5 Preliminary results

The selective search [191] algorithm used for object proposal can extract the candidate object regions by processing an input image under different color spaces. An evaluation of the impact of each color space is here presented, with the intent of establishing the representation that best exploits the characteristics of logo instances, including the already-mentioned high contrast and bright colors. To this extent, the Intersection Over Union (IoU) measure as proposed by Hosang et al. [97] can be used to assess the overall effectiveness of the object proposal. IoU is computed between the ground truth and the proposal as a simple ratio between pixel areas. By varying a rejection threshold, then, the resulting recall is computed in terms of the number of overlapping bounding-boxes. Note that this measure does not take into account the precision component, as false positive examples are expected to be handled by a subsequent classification module. Figure 6.3(a) shows the resulting curves of five possible color spaces, including: HSV, Lab, rg + Intensity, Hue only, and Intensity only. As described in [97], the most significant range for evaluation is between 0.70 and 0.80 IoU. From these experiments, the best



Fig. 6.2 Partial representation of the variability obtained with synthetic data augmentation. Involved transformations include: translation, scale, x-axis shear and y-axis shear.

performing configuration appears to be HSV, suggesting that an explicit representation of saturation and value can successfully exploit the intrinsic features of logo design.

Recognition experiments were performed according to the official criteria defined with the FlickrLogos-32 dataset [168], which is split into three disjoint subsets  $P_1$ ,  $P_2$ , and  $P_3$  as reported in Table 6.1. The system was trained on set  $P_1$ , and validated on  $P_2$  for hyperparameters selection with a target precision of 98% for better comparison with the state of the art [168, 164, 167]. The selected hyperparameters were then used for testing on  $P_3$ .

Figure 6.3(b) shows the performance level obtained on both validation set and test set in terms of precision and recall. Results obtained using query expansion on the test set are also reported.

Table 6.2 reports a comparison with other approaches. [168] and [167] are based on the bundling of neighboring SIFT-based visual words into unique signatures. [164] uses a statistical model for identifying incorrect detections output by keypoint-based matching algorithms. Results show how, despite reaching reasonable performance levels, keypoint-based approaches are still better than what was obtained by this preliminary investigation, based on SVM classification with deep features.

## 6.1 Preliminary investigation

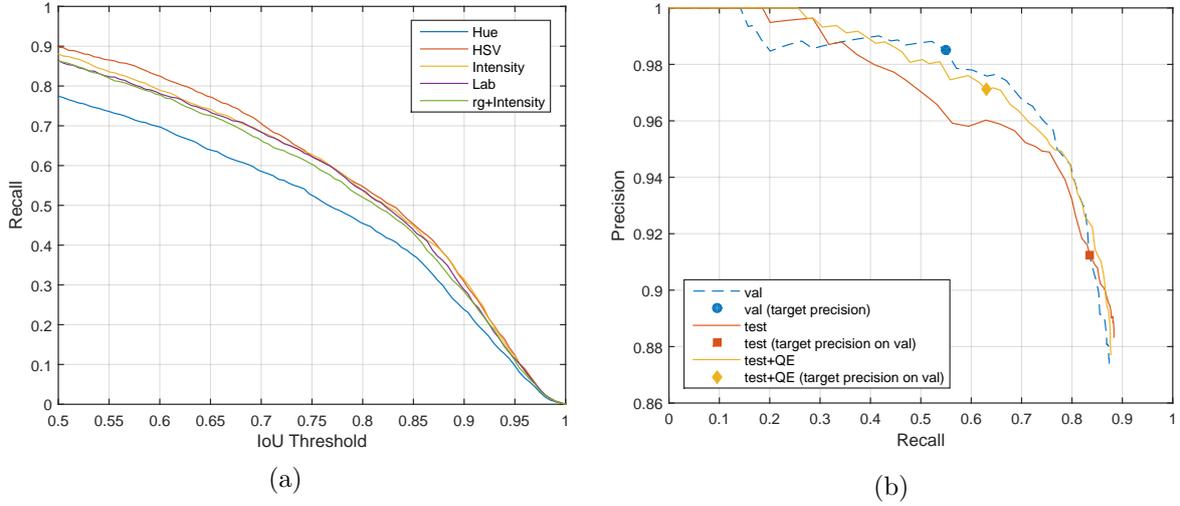


Fig. 6.3 Preliminary results on the Flickr-Logos dataset. (a) Recall vs. IoU with selective search under different color spaces. (b) Precision vs. Recall on validation set and test set (with and without Query Expansion).

Table 6.1 Official partitions of the FlickrLogos-32 dataset.

| Partition              | Description   | Images               | Total       |
|------------------------|---|----------------------|-------------|
| $P_1$ (training set)   | Single logo images                                  | 10 per class         | 320 images  |
| $P_2$ (validation set) | Images showing at least one logo<br>Non-logo images | 30 per class<br>3000 | 3960 images |
| $P_3$ (test set)       | Images showing at least one logo<br>Non-logo images | 30 per class<br>3000 | 3960 images |

Table 6.2 Performance comparison with other approaches.

| Method                                    | Precision   | Recall |
|---|-------------|--------|
| Deep features (preliminary investigation) | 0.91        | 0.84   |
| Deep features + Query expansion           | 0.97        | 0.63   |
| Spatial layout encoding [168]             | 0.98        | 0.61   |
| Noisy score rescaling [164]               | $\geq 0.98$ | 0.73   |
| Feature neighborhood aggregation [167]    | 0.999       | 0.832  |

## 6.2 Proposed approach

The preliminary study should have provided a sense of the scale and difficulty of logo recognition. Despite evidence of the great efficacy of deep learning solutions to classification [113], a basic deep-feature-based approach was shown to still have room for improvement. The presented solution was not relying completely on deep learning: the final classification was in fact performed via support vector machine. The main reason behind this choice, was a substantial unbalancing between network parameters (on the order of 60 Millions) and available data (about 1000 labeled positive examples for training).

It has been shown that balancing more carefully data and parameters can be a key element for the successful training of a deep neural network [162]. Synthetic data augmentation as performed in the preliminary investigation was the first step in this direction, although not sufficient to completely solve the unbalancing problem. For this reason, a new dataset called Logos32-plus has been collected and annotated, extending tenfolds the cardinality of the existing standard in the state of the art. The second step towards a better balancing naturally involves the neural network architecture itself. To this extent, a shallower model has been designed, and a set of smarter strategies from literature have been integrated into the training process, including class balancing, sample weighting, and contrast normalization.

This section describes in more detail the outlined approach, based on an improved training set, a shallower neural model, and the adoption of smarter training strategies.

### 6.2.1 Logos-32plus dataset

The Logos-32plus dataset has been created as a direct extension of the original FlickrLogos-32: it contains the same 32 logo classes, but with an average of 400 examples per class. Each image contains one or more instances of the same logo class, for a total of 12302 logo instances among 7830 images. The precise cardinality distribution is shown in Table 6.3 and Figure 6.4.

As the original Flickr-Logos32 dataset was collected aiming to a keypoint-based approach, the set of images it is composed of was implicitly filtered in order to contain very well-defined instances, with little to no noise or blurring. The main variability was given instead by perspective distortions, which are well-handled by the aforementioned techniques. The new Logos-32plus dataset has been created with the explicit intention of overcoming this limitation, therefore including a wide set of real-life conditions for distortion. An initial selection of images was collected by automating a Flickr and Google

## 6.2 Proposed approach

Table 6.3 Details comparison of 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 per class (Train + Validation) | 40             | 400          |
| Total annotations                                  | 3405           | 12302        |

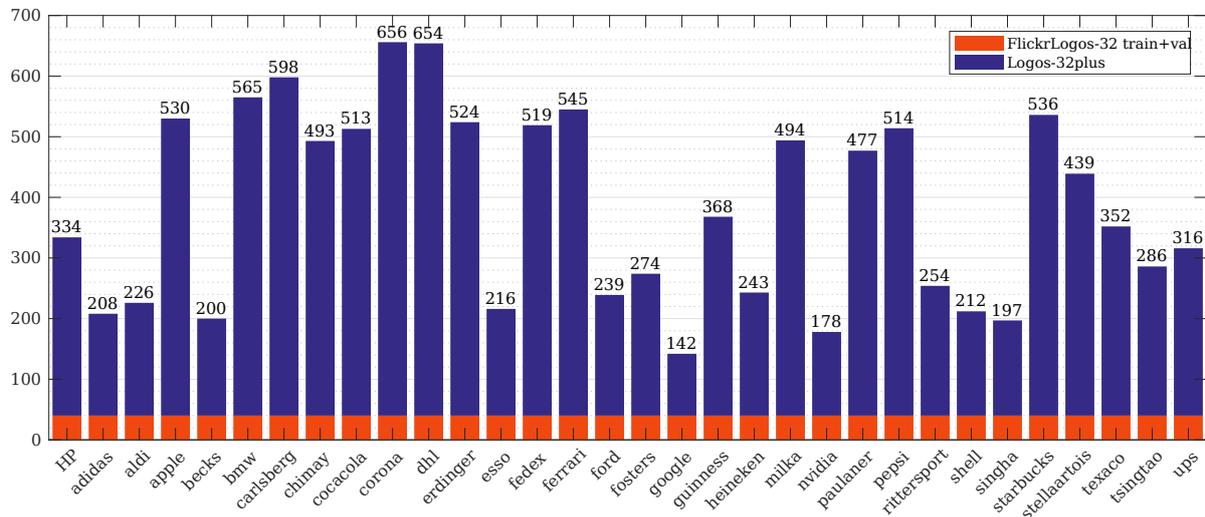


Fig. 6.4 Graphical comparison of the distribution of the 32 logo classes between FlickrLogos-32 (40 examples per class) and the augmented Logos-32plus dataset.



Fig. 6.5 (a) Simple query results. (b) Extended query results (Fedex plane, Fedex truck, Fedex parcel, ...)

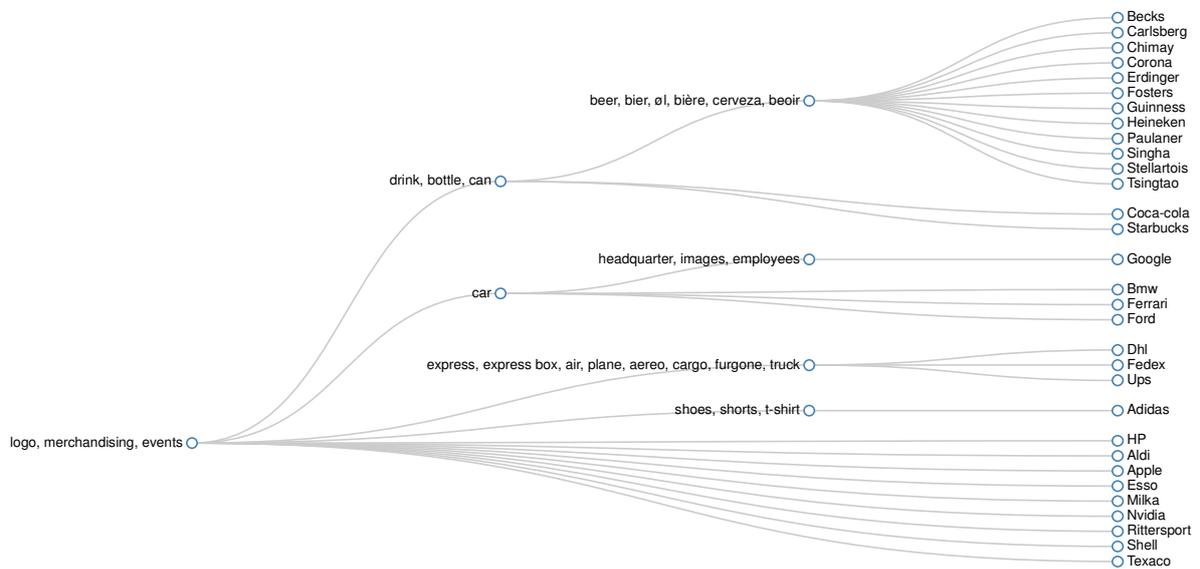


Fig. 6.6 Structure of shared keywords among different logo classes, used to collect images for the Logos-32plus dataset. For example, the extended query set for BMW is composed of: “BMW logo”, “BMW merchandasing”, “BMW events”, “BMW car”.

Images search on the 32 logos classes. However, a basic name search resulted in extremely low variability in the type of results, as shown in Figure 6.5. An extended query set was therefore manually compiled, in order to expand the diversity and cardinality of images by integrating meaningful keywords, for example “Fedex logo” can be extended with “Fedex plane”, “Fedex truck”, “Fedex parcel” and so on. The dendrogram in Figure 6.6 shows the structure of shared keywords among different logos, since several classes have common traits. For example, “UPS” and “DHL” reasonably share the extended keyword set with Fedex, as they all belong to the same brand category (delivery company).

**Near-duplicates removal** A refinement phase was then conducted on the gathered images in order to ensure proper diversification among the new images, as well as to guarantee that no overlap existed between the presented Logos-32plus dataset and the original FlickrLogos-32. This objective was achieved in two steps: a fully-automatic search and a semi-supervised one.

A first quick skimming was performed using the Structured Similarity Measure (SSIM) [196] to compare each image pair in Logos-32plus, as well as each pair between Logos-32plus and FlickrLogos-32. All image pairs with SSIM value equal to or greater than 0.9 were automatically removed.

Secondly, the following procedure was applied in order to obtain near-duplicates removal:

- A neural model was defined according to the criteria described in Section 6.2.2, and trained from scratch on the current dataset.
- Deep features were extracted from each logo instance of both datasets using the newly-trained model (activations of the first Fully-Connected layer).
- A 5-Nearest-Neighbors classifier was populated with such deep features, and used to retrieve the five nearest instances by looking at both the original and the new dataset.
- The 5 nearest neighbors were visually compared with the corresponding source in order to manually exclude any possible remaining duplicate.

This careful refinement phase allowed the removal of near-duplicates like the one depicted in Figure 6.7, where the whole image is overall different, while the specific logo subregion is significantly similar.



Fig. 6.7 Example of near-duplicates detected by the developed procedure. The two logo instances (“Esso”) are extremely similar, while the overall appearance of the two images differs significantly.

Table 6.4 Architecture of the shallower neural network used for logo classification.

| N. | Layer type      | Characteristics        |
|----|-----------------|------------------------|
| 1  | Convolution     | $32 \times 5 \times 5$ |
| 2  | Max Pooling     | Stride 2               |
| 3  | ReLU            | -                      |
| 4  | Convolutional   | $32 \times 5 \times 5$ |
| 5  | ReLU            | -                      |
| 6  | Avg Pooling     | Stride 2               |
| 7  | Convolutional   | $64 \times 5 \times 5$ |
| 8  | ReLU            | -                      |
| 9  | Avg Pooling     | Stride 2               |
| 10 | Fully Connected | 64                     |
| 11 | ReLU            | -                      |
| 12 | Fully Connected | 33                     |
| 13 | Softmax         | -                      |

### 6.2.2 Shallower network

After the extension of labeled training data as described in the previous section, the second step towards a better balancing of data and parameters is to intervene on the neural structure itself. The structure adopted here is inspired by what was used by Krizhevsky et al. [113] on the CIFAR-10 dataset, performing object classification on 10 classes from  $32 \times 32$  pixels RGB images. This extremely shallow network, visualized in Table 6.4, is composed of three Convolution-ReLU-Pooling blocks. Each pooling layer reduces to half the spatial resolution of the input activation. The final layers are two Fully-Connected layers, followed by softmax for classification.

This architecture is composed of roughly  $1.5 \times 10^5$  parameters, compared to the  $6 \times 10^7$  parameters of the previously adopted AlexNet structure [113]. An additional benefit of employing a network with fewer parameters is the faster computation time, which gives the possibility of multiplying the experiments in the search of the best training configuration. Furthermore, with better balancing there is no need for regularization strategies such as dropout [184] and dropconnect [192], as the risk of overfitting is significantly lower [162].

### 6.2.3 Smarter training strategies

Special attention was given to the selection of both positive and negative examples:

- Training on object proposals: The positive examples used to train each class include the manually-labeled ground truth, as well as the regions generated by the object proposal that overlap with the ground truth by a significant amount (Intersection Over Union  $\geq 0.5$ ). The misalignment of these regions with respect to the corresponding logo instance is here theorized to be biased by both the object proposal algorithm in general and the type of logo specifically. A similar misalignment is therefore supposed to be found at test time, hence the advantage of training on similarly-misaligned examples.
- Background class: In addition to the 32 logo classes, a 33rd reject class is explicitly introduced in the model. The corresponding background examples are not selected in a completely random fashion, instead they are sampled from those object proposals that do not overlap with any logo in the training set. This is once again supposed to exploit the hypothesized bias also found at test time, thus improving the discrimination between logo and background.

Various strategies have been applied with the intent of creating a more robust training process. Each one has been singularly evaluated, in order to determine its effective impact on the overall recognition process. These strategies include:

- Synthetic data augmentation: Each training location is perturbed by randomly introducing a shifting misplacement on both the vertical and horizontal axis. This effect is introduced online by a different amount every time a training example is loaded into memory, expanding the variations at training time indefinitely. The joint effect of real and synthetic data augmentation is visually shown in Figure 6.8
- Sample weighting: The contribution of each predicted example on the global loss is weighted proportionally to the overlap between the object proposal and the corresponding ground truth: if the located object proposal is highly misaligned with respect to the ground truth annotation, it can be assumed that the logo is not well visible. In this case, a less confident prediction can be considered acceptable, hence the imposed lower impact on the global loss.
- Class balancing: Logo classes of the newly-collected Logos-32plus have different cardinalities. Class balancing has been implemented by virtually replicating examples of classes with lower cardinality, according to two possible strategies:
  - Epoch-wise: classes are balanced with respect to the whole training epoch.
  - Batch-wise: classes are (also) balanced with respect to each training batch.



Fig. 6.8 Visualization of the joint contribution of both real data augmentation (obtained through the collection of the Logos-32plus dataset) and synthetic data augmentation (obtained with random translation).

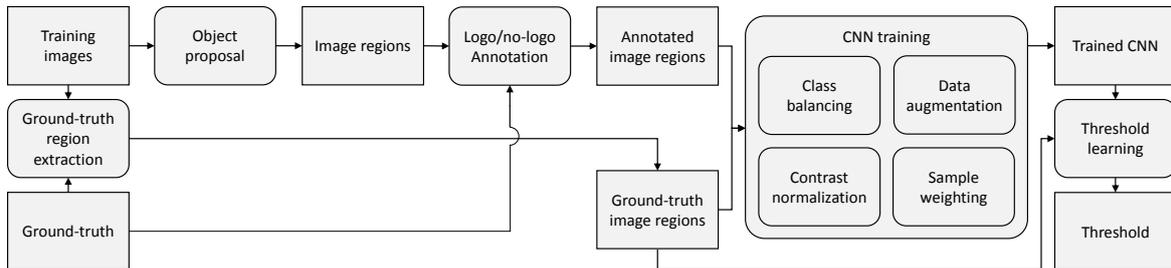


Fig. 6.9 Detailed training procedure developed for logo recognition.

- Contrast normalization: Images are contrast-normalized by subtracting the mean and dividing by the standard deviation of the whole training set. This was shown to make neural models more robust with respect to changes in imaging conditions [155].

Finally, a threshold has been used to determine if a given input is a logo, or not, based on the confidence of the maximum score given by the neural model. This threshold has been automatically tuned to maximize the accuracy on FlickrLogos-32 training and validation sets. The complete pipeline is shown in Figure 6.9

## 6.3 Results

Given a test image, object proposals based on selective search were first generated. Each proposal was then subject to contrast normalization, if used at training time, before being input to the trained neural model. The predicted class confidence scores on all object proposals from the single input image were pooled by selecting the one with the highest confidence, excluding the prediction on background class. If an object proposal captures a non-logo region, in fact, its subsequent classification into “background” class should not determine the overall presence or absence of a logo in the whole image. The highest-scoring logo class is instead compared to the learned reject threshold, in order to

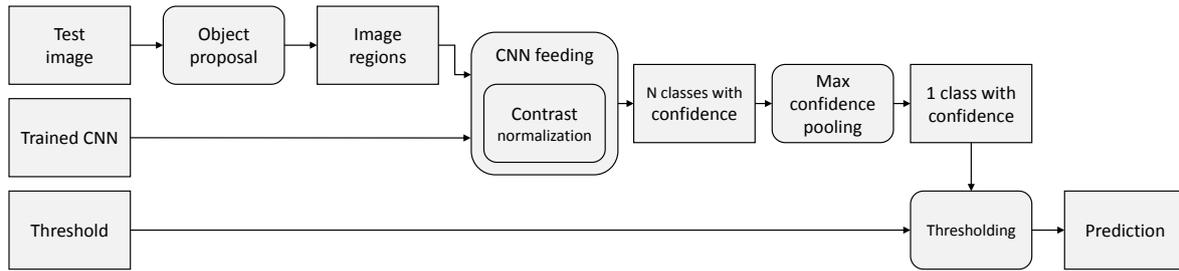


Fig. 6.10 Complete testing procedure applied for logo recognition.

assign a final class to the whole image. The overall test framework is shown in Figure 6.10.

Each modification to the training procedure has been introduced one at a time, in order to evaluate its individual contribution, and the corresponding value is underlined in Table 6.5 for better readability. The “Data augmentation” column refers to synthetic data augmentation: all configurations have in fact always been evaluated with real data augmentation, derived by extending the FlickrLogos-32 training and validation set with the newly collected Logos-32plus. Results are shown in Table 6.5 in terms of both F1 and accuracy on the official FlickrLogos-32 test set.

The results presented in Table 6.5 show that, compared to a direct deep learning application to the logo recognition task (e.g. Training Configuration I, for brevity *TC-I*), the different considered training choices can produce a noticeable increase in performance:

- The first leap in performance was achieved by including the 33rd background class. The results (*TC-II*) show F1 to improve by 31.8% and accuracy by 52.4%, compared to *TC-I*.
- A second jump was obtained by including object proposals coming from selective search as examples of additional training. This is configuration *TC-III*, and it improves the F1 measure and accuracy by 11.3% and 12.4% respectively, compared to *TC-II*.
- A third leap in terms of performance was obtained by synthetic data augmentation, i.e. increasing the cardinality of object proposals by perturbing them with random translations. This configuration (*TC-IV*) improves the F1 measure and accuracy of 11.7% and 20.9% compared to *TC-III*.
- A smaller improvement in performance was then achieved by including class-balancing, in order to take into account the different cardinalities. The epoch-wise balancing offers better performance compared to its batch-wise counterpart

Table 6.5 Analysis on the contribution of each training choice on global performance values (Precision, Recall, F1, and Accuracy). Each change in training configuration is underlined for readability, and the final best configuration is highlighted in boldface.

| TC         | BG class   | Pos. exam.   | Data augm. | Class balanc. | 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        | <u>GT+OP</u> | <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        |
| <b>VII</b> | <b>Yes</b> | <b>GT+OP</b> | <b>Yes</b> | <b>Epoch</b>  | <b>Yes</b>   | <b>No</b>     | <b>0.989</b> | <b>0.906</b> | <b>0.946</b> | <b>0.958</b> |
| 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>    | <b>0.989</b> | 0.866        | 0.923        | 0.955        |

Legend to Table 6.5

|                      |  |
|----------------------|--|
| <b>TC</b>            | Identifier for the Training Configuration                              |
| <b>BG class</b>      | Background class (no-logo examples) included in training               |
| <b>Pos exam.</b>     | Sources for positive training examples                                 |
|                      | <b>GT</b> Precise ground-truth logo annotations                        |
|                      | <b>GT+OP</b> Precise ground-truth and Object-proposal logo annotations |
| <b>Data Augm.</b>    | Data Augmentation (translation)  |
| <b>Class balanc.</b> | Class balancing to account for different cardinalities                 |
|                      | <b>Epoch</b> Classes are balanced in each epoch                        |
|                      | <b>Batch</b> Classes are balanced in each batch as well                |
| <b>Contr. norm.</b>  | Pre-processing of training examples with contrast normalization        |
| <b>Sample weight</b> | Weighting examples based on overlap between OP and GT                  |

(respectively *TC-V* and *TC-VI*). In particular, *TC-V* improves the F1 and accuracy by 0.4% and 0.3% compared to *TC-IV*.

- Contrast normalization brought a further small, but consistent, improvement, with *TC-VII* improving the F1 measure by 2.4% and accuracy by 0.2% compared to *TC-V*.
- Sample weighting (in both *TC-VIII* and *TC-X*), which determines a loss value proportional to the overlap between object proposal and corresponding ground truth, appears to negatively impact the overall method performance.

The best configuration (i.e. *TC-VII*) trained on the extended training set is highlighted with boldface in Table 6.5 and compared with the state of the art in Table 6.6. Performance values for other methods are taken from the respective papers, hence some of the measures are missing. From the reported results it is possible to see that the proposed solution, based on the rebalancing given by extended training set and reduced network complexity, is able to improve the F1 measure with respect to the best state of the art method of 3.8%.

As a further comparison, the last row of Table 6.6 presents results obtained using only FlickrLogos-32 (training and validation set) for training and keeping unchanged all other training choices. This translates in a drop of 14.7% in F1 measure and 4.8% in accuracy, providing a concrete measure of the role of real data augmentation, compared to a purely synthetic one [62].

A final experiment was conducted in order to determine whether the main source of error comes from the selective search (supposedly not able to reach high enough recall), or from the neural model itself (providing a wrong classification of correctly detected logo regions). This was done by including the actual ground truth regions among the object proposals, bypassing the hypothetical limits of the selective search. The resulting performance is characterized by only a slight improvement (F1 measure by 0.2% and accuracy by 0.2%), therefore indicating that the main source of error lies in the neural model itself.

Figure 6.11 shows some examples of wrongly-classified logos. The candidate regions were proposed by the selective search algorithm, and are here filtered to display only those having Intersection over Union (IoU) greater than or equal to 0.5 with respect to the corresponding ground truth. For each mistaken logo, the nearest neighbor from the training set is also shown below, in order to provide some visual insights on the main sources of classification error. It is possible to observe, for example, a noticeable tendency

Table 6.6 Best training configuration from Table 6.5 compared with other methods in the state of the art.

| Method                      | Training data | Precision    | Recall       | F1                 | Accuracy     |
|-----------------------------|---------------|--------------|--------------|--------------------|--------------|
| BoW SIFT [167]              | FL32          | 0.991        | 0.784        | 0.875              | 0.941        |
| Romberg et al. [168]        | FL32          | 0.981        | 0.610        | 0.752              | N/A          |
| Revaud et al. [164]         | FL32          | $\geq 0.980$ | 0.726        | $0.834 \div 0.841$ | N/A          |
| Romberg et al. [167]        | FL32          | <b>0.999</b> | 0.832        | 0.908              | N/A          |
| Bianco et al. [15]          | FL32          | 0.909        | 0.845        | 0.876              | 0.884        |
| Bianco et al. + Q.E. [15]   | FL32          | 0.971        | 0.629        | 0.763              | 0.904        |
| Eggert et al. [62]          | FL32          | 0.996        | 0.786        | 0.879              | 0.846        |
| Oliveira et al. [149]       | FL32          | 0.955        | <b>0.908</b> | 0.931              | N/A          |
| DeepLogo [102]              | FL32          | N/A          | N/A          | N/A                | 0.896        |
| This work ( <i>TC-VII</i> ) | FL32, L32+    | 0.989        | 0.906        | <b>0.946</b>       | <b>0.958</b> |
| This work ( <i>TC-VII</i> ) | FL32          | 0.976        | 0.676        | 0.799              | 0.910        |

for the model to overly rely on color features. This is possibly a consequence of the very small input size chosen for the neural structure ( $32 \times 32$  pixels).

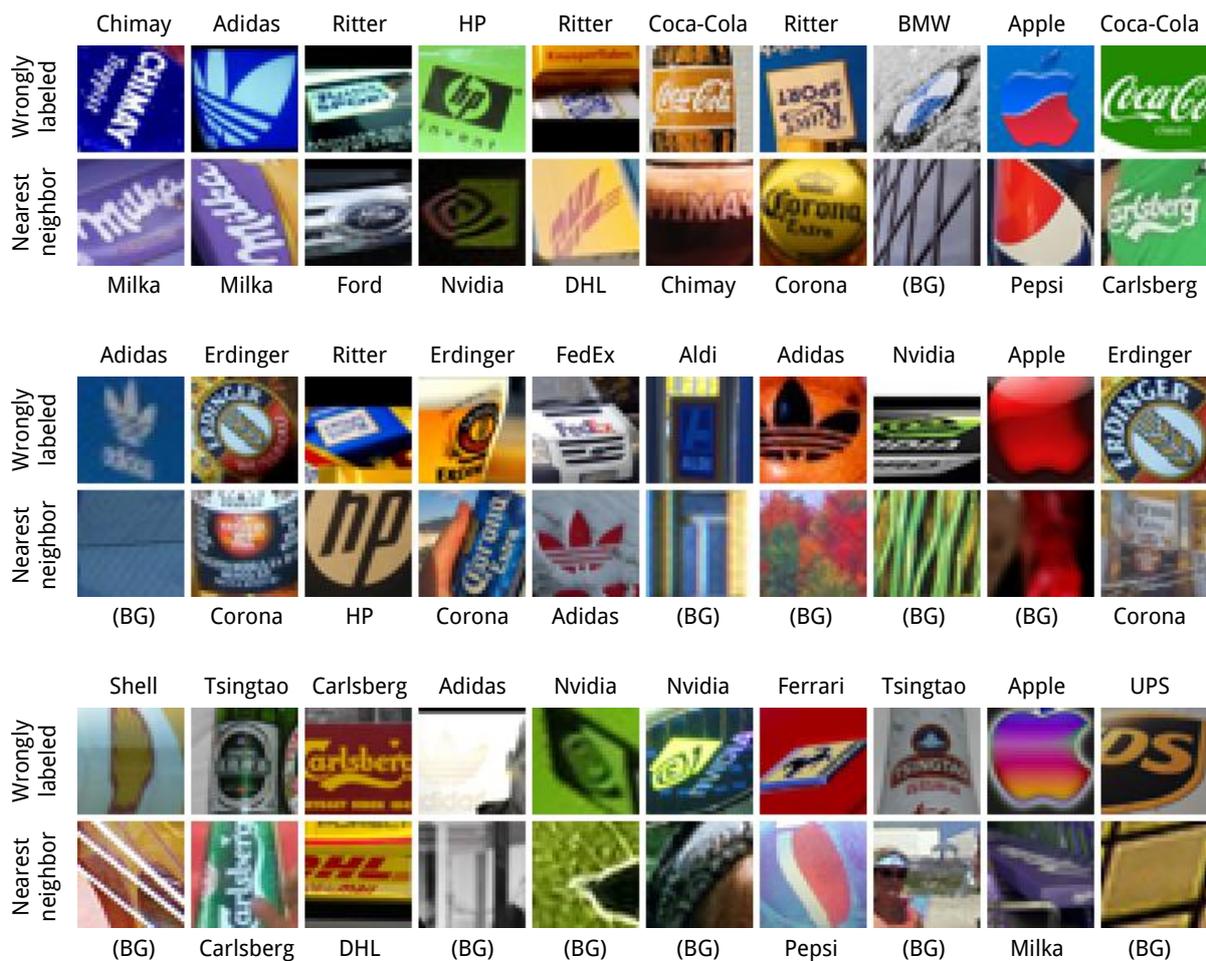


Fig. 6.11 Examples of wrongly-labeled logos, sorted by classification confidence (i.e. the first, top-left, logo is the most confident error made by the neural model). For each mistaken logo, the nearest neighbor from the training set is shown for comparison, using the same description adopted for near-duplicates removal.

# Chapter 7

## Main-subject classification: Vegetables and fruits

*“Pepperoni and green peppers, Mushrooms, olive, chives. Pepperoni and green peppers, Mushrooms, olive, chives.”*

System of a Down - Chic 'n' Stu

Changing the visual classification domain often implies facing different challenges. While logo recognition deals with constrained perspective distortions and classes that are discriminable by design, other classification problems may present more challenging characteristics. Specifically, the focus on this chapter is on fine-grained classification of edible plants, fruits, and mushrooms. This family of problems requires the correct identification of classes that exhibit high inter-class similarity and at the same time low intra-class similarity, as shown in Figure 7.1. As such it depends on the ability to spot very subtle differences that might help in class discrimination.

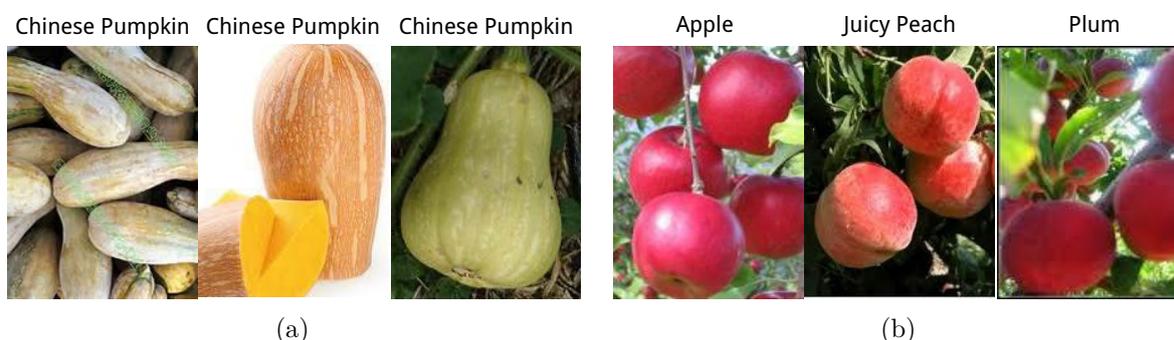


Fig. 7.1 Some of the challenges specific to fine-grained visual classification include a) large intra-class variability and b) small inter-class variability.

Recognition of vegetable, fruits, and food in general, can enable common-life objects to acquire greater value through the introduction of smart capabilities [157]. Based on the recognition of available products, for example, a smart refrigerator would be able to perform tasks that include:

- Proposing recipes. Through the application of global and user-customizable ingredient networks [187], it is possible to suggest recipes that comply with daily diet monitoring schedules [49].
- Analyzing conservation status. Texture-based processing [58] and analysis of features related to cooking states [104] can help tracking the conservation status of stored goods.
- Suggesting shopping items. By combining recommender systems with methods for visual object counting [117], it is possible to provide the user with meaningful hints for items to buy.

All these features can be addressed with, or inspired by, existing literature work. However, the first step in this direction is a proper recognition of the raw food itself.

To this extent, a very recent dataset is exploited, containing vegetables, fruits, and other types of raw food. Classes in this dataset are organized in a hierarchical structure, following standard agricultural taxonomies. The classification task is addressed with different existing neural architectures, eventually finding and exploiting an architecture that outperforms the state of the art. Different types of hierarchical training are investigated and proposed, in order to exploit the taxonomy provided with the dataset and further improve the classification performance. The recognition pipeline is augmented with an object proposal based on salient object detection, which is shown to improve the final recognition performance of selected architectures.

## 7.1 Datasets and related works

The literature offers several food-related datasets, such as UNIMIB2016 [50], Food-101 [32], UNICT889 [67], although very few focus on fruits and vegetables, specifically Fruits 360 [145] and VegFru [100]. VegFru is a dataset for fine-grained visual categorization specific to fruits (92 classes) and vegetables (200 classes, including several non-vegetable “mushroom” classes). Images belonging to each class are also labeled with an additional super-class following official agricultural and horticultural taxonomies, as depicted in Figure 7.2, for a total of respectively 10 and 15 super-classes. This hierarchical partitioning

## 7.1 Datasets and related works

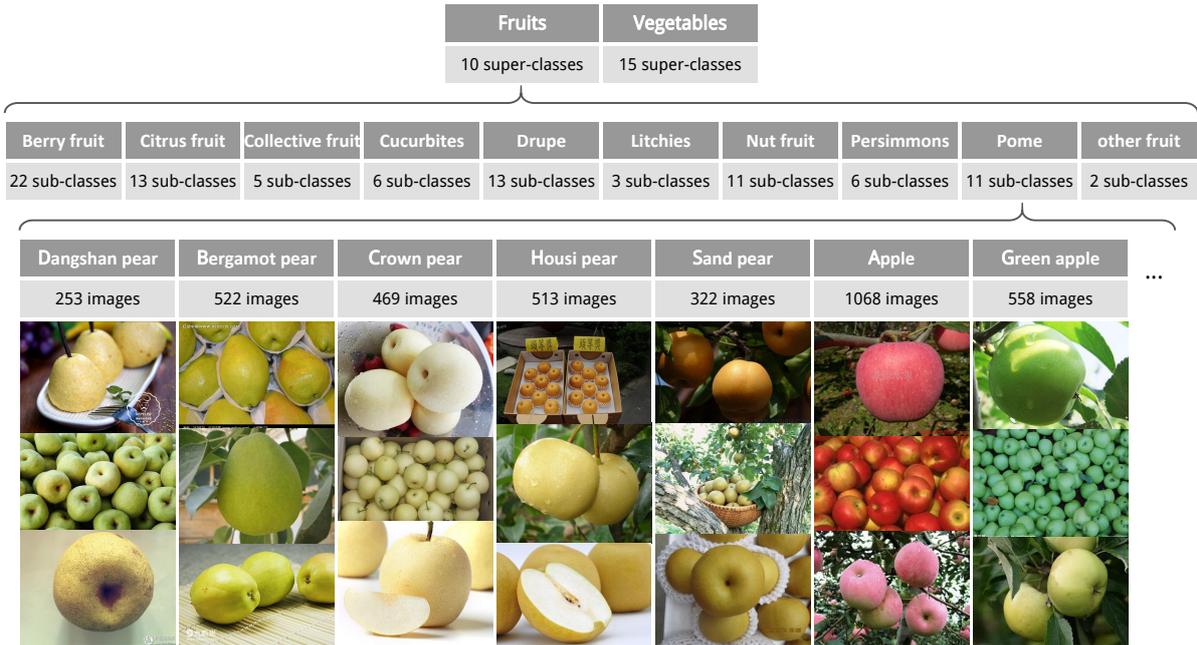


Fig. 7.2 Detail of the hierarchical annotation provided with the VegFru dataset. This figure shows a subset of the Fruit/Pome sub-class, highlighting the subtle interclass differences.

is also used to separate different edible parts of the same item, which are potentially cooked in disparate ways, e.g. “soybean”, “soybean sprout”, and “soybean seed”. This dataset was selected for the training of a vegetable recognition module, due to its high cardinality (more than 160000 images), and variety of acquisition conditions.

In [100] authors trained a neural network architecture based on VGG-16[181] (from Visual Geometry Group) to address both the sub-class and super-class problem of VegFru. They showed how it is possible to improve performance on the sub-classes, which is inherently more challenging, by fusing features extracted through the processing of both problems with a Compact Bilinear Pooling layer (CBP) [76]. In this chapter, other fusion strategies are proposed and evaluated, exploiting the available hierarchical labeling. The VegFru dataset is very recent, for this reason no other methods have yet been published that tackle its specific classification problem. A subset of this dataset is also used in Chapter 9 as an auxiliary task to latently train a color constancy neural network without the need for explicit illuminant annotations, which was made possible thanks to the importance of color in vegetable discrimination.

## 7.2 Outline of vegetable and fruit recognition

This section presents the proposed approach to automated recognition of edible fruits and vegetables. The efforts on exploiting the hierarchical nature of such classes can however be easily extended to a broader set of food types and classes in general.

### 7.2.1 Neural baseline architectures

In [100] authors presented a baseline evaluation of several off-the-shelf neural networks on the VegFru dataset, including AlexNet [113], GoogLeNet[186], and VGG-16[181]. They report best results on the use of GoogLeNet, followed closely by VGG-16. The analysis offered in [36] and [20] highlights how GoogLeNet and VGG-16 produce comparable results on the ImageNet challenge[113], with VGG-16 requiring a much larger set of parameters ( $\sim 125$  millions vs.  $\sim 10$  millions). A promising alternative suggested by the same study appears to be the ResNet architecture [92]: in particular, ResNet-34 and ResNet-50 produced a 5% performance leap compared to GoogLeNet, at the expense of only double the number of parameters ( $\sim 20$  millions). For this reason, both ResNet-34 and ResNet-50 are evaluated in the following in terms of accuracy on the task of vegetable and fruit recognition, by fine-tuning and testing such architectures.

Authors of [221] developed a training framework that learns the appropriate neural architecture directly from observing the training data. They were able to automatically design (and, subsequently, train) state of the art architectures on a multiplicity of tasks, as well as prove the generalization capability of the learned architectures on tasks that were not seen during the search for an optimal neural structure. This study is here extended by exploiting a network that was automatically designed for object recognition on CIFAR-10 [112], which will be referred to as NASNet (from Network Architecture Search).

### 7.2.2 Hierarchical labeling

The VegFru dataset is supplied with a two-level-hierarchy labeling, as described in Section 7.1. Authors of [100] demonstrated the advantage of exploiting top-level annotations (so-called “super-classes”) to improve the bottom-level predictions (or “sub-classes”), which was shown to be a more challenging task. This section describes two techniques to exploit hierarchical labeling through the specialization of fully-connected layers.

The last block of a Convolutional Neural Network (CNN) for classification is typically characterized by a fully-connected layer, i.e. a linear combination mapping an  $N$ -

dimensional feature vector to the representation of the final problem size. This is also preceded by other fully-connected blocks, building an internal hierarchy of interpretation used by the neural network. The main idea is to infuse this implicit representation with information coming from the explicit multilevel annotation, as depicted in Figure 7.3:

1. The selected neural architecture is trained (or fine-tuned) on the super-classes problem (lower cardinality).
2. The whole model is further fine-tuned on the sub-classes problem, using one of the following approaches:
  - Appending to the existing final fully-connected layer a new one.
  - Replacing the final fully-connected layer.

In both cases, the final neural network will take as input the image, and produce a prediction vector with the same dimensions as the number of sub-classes.

### 7.2.3 Saliency-based proposal

Chapter 3 presented the problem of automatically inferring the most salient element of an image, by exploiting semantic middle-level features without relying on a class-specific training. Here the concept of salient object detection is effectively tested as an object proposal step, by automatically cropping the input region around the depicted fruit or vegetable, before performing the classification step itself. Specifically for the experiments, state of the art DSS algorithm [99] is here employed.

The generated saliency map is first binarized with an adaptive global threshold [150], and the tightest bounding box around pixels set to 1 is used as a first approximation of the autocropping region [7, 52]. A minimum size is set on the proposed bounding box, which is also forced to have a square aspect ratio. This is done in order to avoid upsampling and stretching artifacts when feeding the image to a fixed-size-input network for classification.

## 7.3 Experiments

In this section the proposed solutions are systematically evaluated under the standard evaluation criteria posed by authors of the VegFru dataset [100].

The ResNet34, ResNet50, and NASNet architectures have been fine-tuned on the VegFru training subset, composed of 29200 examples, by applying data augmentation

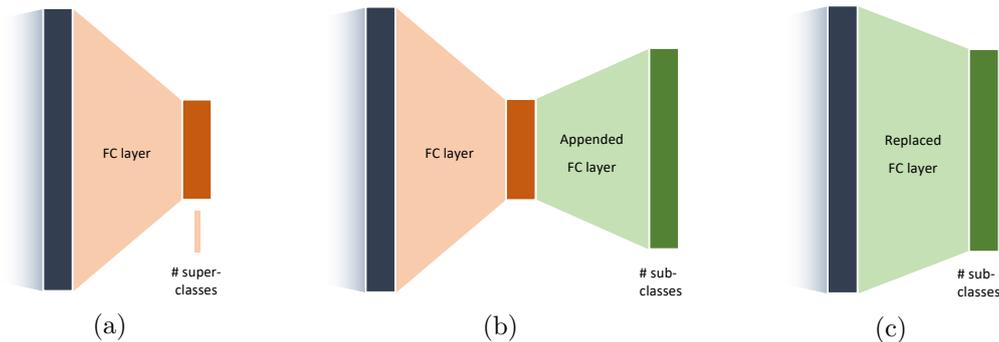


Fig. 7.3 The fully-connected (FC) specialization exploits hierarchical labeling to improve performance on the sub-classes, with respect to directly training on the sub-classes themselves. (a) The initial FC always maps to the number of super-classes. (b) The “append” specialization concatenates a new FC layer. (c) The “replace” specialization substitutes the original layer with a new one.

Table 7.1 Results for baseline architectures on both the super- and sub-class problem. The proposed solutions, based on ResNet and NASNet, outperform the results presented in [100].

| Neural architecture | Super-classes<br>(25 classes) | Sub-classes<br>(292 classes) |
|---------------------|-------------------------------|------------------------------|
| AlexNet             | 72.87%                        | 66.40%                       |
| [100] GoogLeNet     | 82.52%                        | 79.22%                       |
| VGG-16              | 82.45%                        | 77.12%                       |
| This work ResNet-34 | 82.06%                        | 81.14%                       |
| ResNet-50           | 85.33%                        | 79.78%                       |
| NASNet              | <b>87.41%</b>                 | <b>84.50%</b>                |

techniques that include random crop, random rotation, and random horizontal flip. These models, previously trained on the ImageNet task [113], were here fine-tuned via Stochastic Gradient Descent (SGD) with momentum, decaying learning rate of an order of magnitude every 30 epochs starting from 0.01, setting batch size at 4, for a total of 100 epochs. The epoch that produced the best accuracy on the validation set (14600 images) was then selected for evaluation, in order to avoid overfitting both on the training and final test data. The models were evaluated on the VegFru test set (116931 images) using mean top-1 accuracy, i.e. accuracy is computed for each class separately, and subsequently averaged. Results are presented in Table 7.1.

Coherently with the analysis reported in Section 7.2.1, on the sub-class problem ResNet-34 performed better than all previously investigated baselines [100], while ResNet-

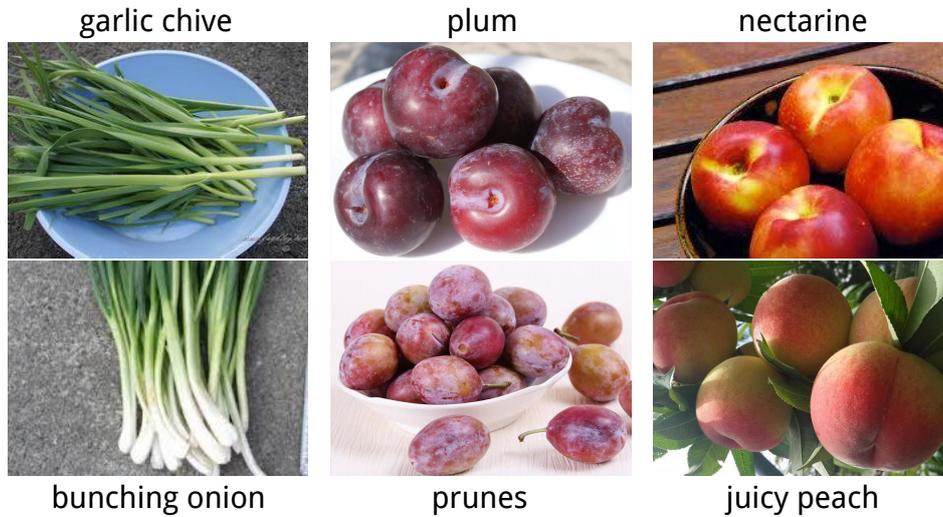


Fig. 7.4 Often-confused class pairs for the best performing model, based on NASNet. The mistakes are mainly due to extreme similarities between involved classes.

50 improved performance on the super-class problem. For both tasks, however, the best results were obtained through the fine-tuning and application of NASNet, respectively reaching 87.41% accuracy on the super-classes problem, and 84.50% on the sub-classes. This solution, based on a single neural model, outperformed also the 83.51% accuracy obtained by authors of [100] on VGG-16, who used Compact Bilinear Pooling [76] to merge the features of two neural networks independently trained on the super- and sub-class task. Example mistakes of the NASNet architecture that was fine-tuned to the task of vegetable and fruit recognition are shown in Figure 7.4.

The second set of experiments aims at assessing how hierarchical label exploitation impacts the recognition performance on the sub-classes problem. Following Section 7.2.2, ResNet-34 has been first fine-tuned on the super-class problem, and subsequently applied fully-connected specialization on the sub-classes problem, by either replacing the last layer or appending a new one to it. Results reported in Table 7.2 show that the “append” technique highly degraded the final performance, while “replace” produced an appreciable improvement. This could be explained by the fact that, by constraining the network activations to a lower dimensionality compared to the final problem size. This procedure, in fact, involves a compression of data with a possible loss of information. The promising “replace” technique was then applied to the ResNet-50 and NASNet models as well. While the ResNet-50 architecture gained the most from this specialization, NASNet showed essentially no improvement. This behavior possibly indicates that such a powerful neural

Table 7.2 Results for hierarchical labeling exploitation on the sub-classes problem.

| Neural architecture | Specialization technique | Before specialization | After specialization |
|---------------------|--------------------------|-----------------------|----------------------|
| ResNet-34           | append                   | 81.14%                | 74.93% (−6.21%)      |
| ResNet-34           | replace                  | 81.14%                | 82.06% (+0.92%)      |
| ResNet-50           | replace                  | 79.78%                | 84.10% (+4.32%)      |
| NASNet              | replace                  | 84.50%                | 84.08% (−0.42%)      |

Table 7.3 Impact of saliency-driven object proposal on tested neural architectures. Both ResNet variants exhibit a small, yet consistent, improvement. On the contrary, NASNet shows degraded performance.

|           | Super-classes<br>(25 classes) | Sub-classes (292 classes) |                       |
|-----------|-------------------------------|---------------------------|-----------------------|
|           |                               | Direct training           | Hierarchical labeling |
| ResNet-34 | 82.20% (+0.14%)               | 81.19% (+0.05%)           | 82.41% (+0.35%)       |
| ResNet-50 | 85.91% (+0.58%)               | 80.12% (+0.34%)           | 84.73% (+0.63%)       |
| NASNet    | 86.84% (−0.57%)               | 82.17% (−2.33%)           | 80.57% (−3.51%)       |

model is already able to infer most of the hierarchy-related correlations directly from data annotated with only the sub-classes labels.

Saliency-driven object proposal has been applied to automatically crop the test set of VegFru before performing the actual classification. The impact of this preprocessing step is detailed in Table 7.3: there is a subtle, yet consistent improvement on both ResNet architectures. At the same time, NASNet appears to lose discriminative power, coherently with the behavior observed following fully-connected specialization. The hypothesis is that, for particularly challenging images, the NASNet model had learned to infer the correct class through an analysis of the whole context (for example class-specific leaves and foliage). By excluding this information, the model is “brought back” on par with ResNet-50. In general, it can be stated that the object proposal phase produces a mixed outcome. Some further insights are provided in Figure 7.5: while saliency estimation correctly excludes the background lemon in the left-most image, the right one shows how the saliency model’s implicit bias towards people may inadvertently bring the focus to the wrong subject.

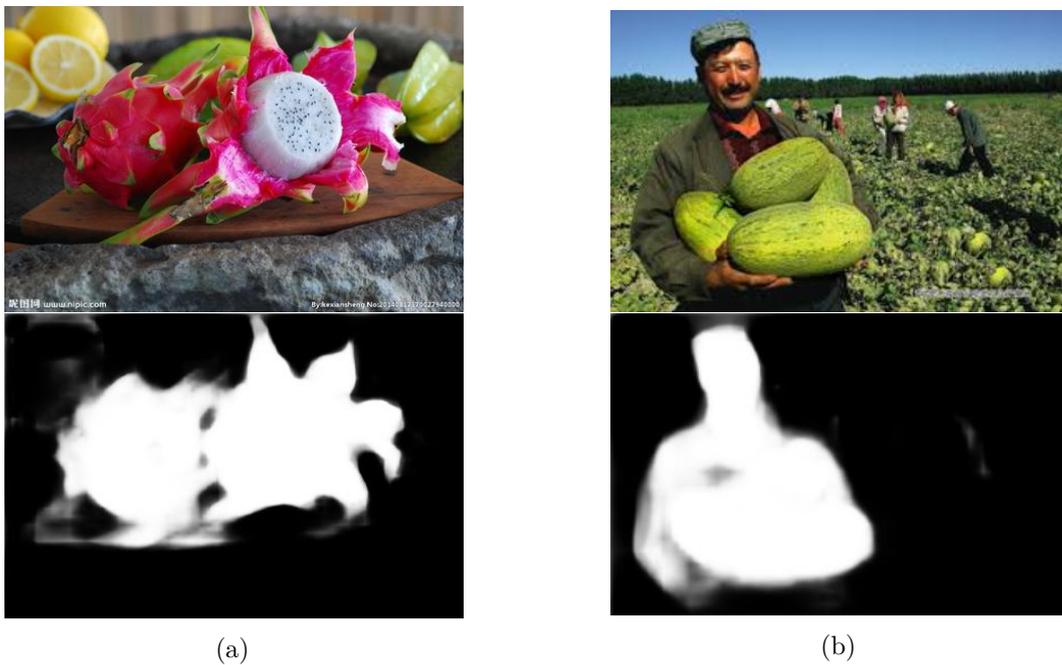


Fig. 7.5 Saliency-driven object proposal for example images in the VegFru dataset. a) The background “distraction” fruits are correctly excluded. b) The human subject is inadvertently included in the saliency estimation.

## **Part III**

# **Attributes extraction**

# Chapter 8

## Introduction to attributes extraction

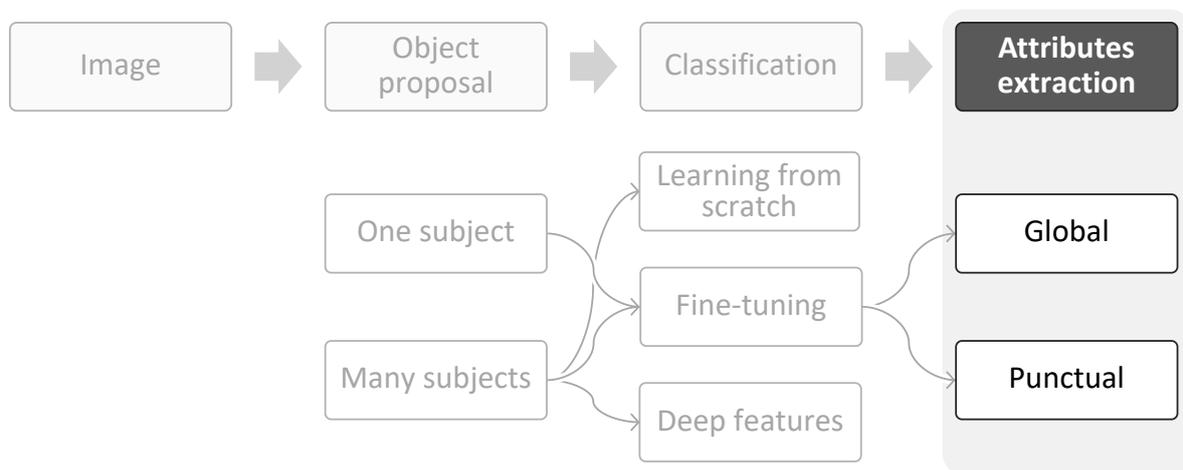


Fig. 8.1 Range of possibilities for the attributes extraction phase.

The final step in the envisioned pipeline for image description is the extraction of numerical attributes from the image itself. This task can benefit from a semantic interpretation of depicted elements: correctly identifying the image content through the previous classification step, in fact, can produce useful pieces of information for a more accurate estimation of image attributes. In particular, the attributes extraction phase can span a whole range of image properties, whose extrema are shown in Figure 8.1:

- Global attributes. These define the whole scene with one characterization, although possibly reached through an analysis of the individual depicted subjects. A global attribute such as the scene illuminant can either be the final goal, as the case with the envisioned general pipeline, or be the preprocessing step for further analysis.

- 
- Punctual attributes. Such attributes have a much finer grain, describing with pixel-precision a given property of the input image.

Intermediate levels of description, such as extraction of region-wise attributes, are also possible. Attributes can, for example, refer to subregions of the image that were identified as belonging to particular classes of interest.

Estimation of the scene illuminant has been selected as the representative example for global attribute extraction [35]. The problem has been addressed through the definition of an original learning strategy that allows an illuminant estimation module to be trained with the objective of optimizing a separate task. Specifically, classification of vegetable classes as seen in Chapter 7 has been exploited as the alternative task to guide the learning process. The advantage of this solution lies in shifting the need for ground truth data to a different problem, for which data collection and annotation is possibly less expensive to perform. The described technique also potentially allows extending the illuminant estimation to a spatially-varying description of the scene, although this goes beyond the scope of this thesis.

As a direct example of punctual attribute extraction, estimation of the distance between the camera and depicted subjects has been investigated [19]. A new alternative representation of pixel-wise distance has been proposed, based on the size in real life of depicted subjects. This data representation is device-agnostic, i.e. it does not depend on any specific camera parameters and settings. As such, it allows for richer models to be trained from heterogeneous datasets, and eventually applied to estimate pixel-level absolute distance information from a single image. For the reported experiments, an architecture similar to the Fully Convolutional Network used in Chapter 3 for salient object detection has been employed. The size values predicted by the model have been quantized into discrete levels with non-linear separations, in order to account the relative effect of misestimating values at long distances. An emerging property common to both solutions is therefore the transformation of a regression problem into a classification one.

# Chapter 9

## Illuminant estimation

*“And the sky is filled with light, can you see it? All the black is really white, if you believe it”*

Nine Inch Nails - In This Twilight

Extraction of meaningful attributes from an image may, directly or indirectly, leverage on classification tasks. A concrete example is estimating the global illuminant, i.e. defining the chromatic properties of the light illuminating the scene. To this extent, the current chapter covers a proposed learning strategy for illuminant estimation that exploits object recognition during the training phase, effectively removing the need for illuminant ground truth datasets. At test time, the trained model can then operate without any requirements for specific classes to be present in the image, as proven by testing on standard datasets for illuminant estimation.

Color constancy is the ability of human beings to recognize the colors of objects independently of the characteristics of the light source. Computational color constancy aims to first estimate the illuminant and subsequently use this information to correct the image, to display how it would appear under a canonical illuminant [71]. The class of “learned” methods is among the most successful illumination estimation methods to date [81, 3], and typically relies on a training set of images which are labeled with the respective scene illuminant. Although the human visual system is often compared to a machine learning algorithm, during evolution it was never presented with ground truth illuminants. Instead, it is hypothesized that the ability of color constancy arose because it helped other crucial tasks, such as recognizing fruits, objects, and animals independently of the scene illuminant [33].

This observation constitutes the main motivation towards investigating to what extent it is possible to learn illuminant estimation as a byproduct of an auxiliary task, such as object recognition. Specifically, recognition of vegetables and fruits will be exploited for

the experimental setup, since color is considered a discriminative feature for such classes, as shown in the following.

## 9.1 Related works

Previous works highlighted the usefulness of object recognition in applications related to color theory, such as assessment of chromatic adaptation transforms through memory color matching [182], performing color constancy with the assistance of faces [26] or via pre-training on generic object recognition before fine-tuning for illuminant estimation [135]. The solution proposed in this chapter is the first machine learning method for illuminant estimation which does not require illuminant annotations, at the same time not relying on specific object presence at inference time.

Other more general approaches to computational color constancy can be divided into “parametric” and “learned”.

### Parametric

Parametric solutions are typically based on handcrafted features which depend on few manually-tunable parameters, the most effective to date being Akbarinia et al. [3] and Cheng et al. [43]. Akbarinia et al. [3] adapted physiological mechanisms to a model based on Gaussian kernels, whose size changes as a function of local contrast variations. Cheng et al. [43] provided an interpretation of why spatial analysis helps in color constancy, and consequently proposed a method that selects pixels that are meaningful for illuminant estimation, using a projection distance in the color distribution.

A complete review of parametric methods for illuminant estimation is available in [81].

### Learned

Learning-based solutions recently proved to be particularly effective when trained in a cross-validation setup: Bianco et al. [23] developed a patch-wise illuminant estimation neural network and generate a global estimation through consensus. Lou et al. [135] propose the fine-tuning of a network that was pre-trained for both object recognition, and illuminant regression based on the predictions of other color constancy methods. Oh et al. [148] reformulate the problem of illuminant estimation as a classification task by clustering the target illuminants. All these solutions require an illuminant ground truth, and are designed to be trained on the same type of images that were seen during the test phase.

## 9.2 Proposed method for illuminant estimation

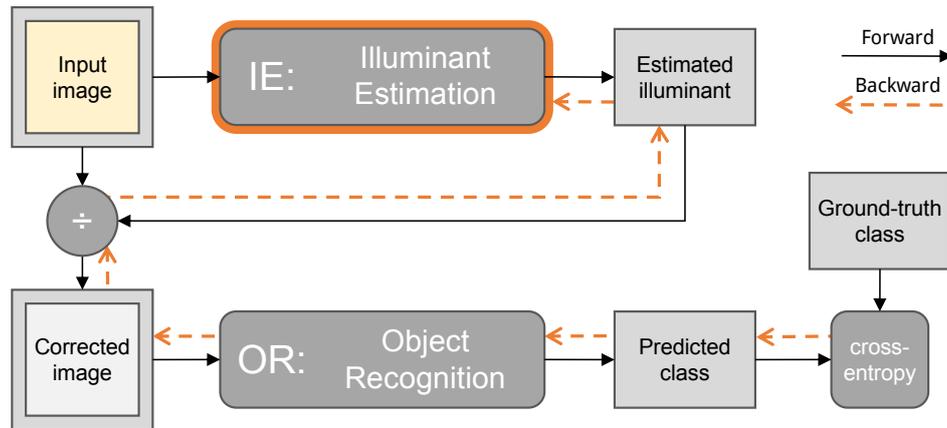


Fig. 9.1 Schematic representation of the proposed learning strategy for computational color constancy. The Illuminant Estimation module is trained with the objective of optimizing the Object Recognition module. At inference time, only the Illuminant Estimation module is needed.

## 9.2 Proposed method for illuminant estimation

This section presents the proposed learning approach to estimate the scene illuminant in the absence of any illumination ground truth data, but with label information for an auxiliary task. Object recognition is here considered as the auxiliary task, but other objectives such as object detection or semantic segmentation could be used as well. The proposed Illuminant Estimation / Object Recognition network (IEOR) is composed of two parts: an Illuminant Estimation module (IE) and an Object Recognition module (OR), as shown in Figure 9.1. Following this structure, IE is learning to predict, for any given image, a color correction that would improve OR.

### 9.2.1 Object Recognition network

The Object Recognition network performs the auxiliary supervised task described in the proposed approach, which is used to indirectly train the Illuminant Estimation network. OR takes an input RGB image, and produces a prediction of class presence. For the purpose of the experiments described in this chapter, the focus will be on vegetable recognition [100], a task for which color is important as highlighted in Figure 9.2. Other problems for which a correct white balancing is expected to improve results might be recognition of painting styles [25], or assessment of image aesthetics [21].

## 9.2 Proposed method for illuminant estimation

---

In order to obtain the desired separation of the network’s internal logic into IE and OR modules, with the intermediate representation being the estimated illuminant, it is necessary to make OR overly-sensitive to color variations. This would then effectively drive IE to work as a useful preprocessing step, i.e. to learn to perform the necessary color correction. To fulfill this requirement, the following strategy is adopted:

1. OR is pre-trained alone on the chosen auxiliary task.  
No color jittering is applied in this phase.
2. IE is connected, and the whole system is trained end-to-end with color jittering as described in Section 9.2.2.  
During this second phase the gradients flow through OR, but only the weights in IE are updated.

If the training process is not constrained as explained, it might lead to trivial or meaningless solutions, for example:

1. IE learns to always output a fixed neutral illuminant, while OR internally learns both color correction and object classification.
2. OR and IE spread the job of estimating illuminant and recognizing the object, without a clear distinction of the intermediate “estimated illuminant”.
3. IE learns some arbitrary 3-number output, (that would generate a non-intelligible image), while OR learns to perform object classification from such representation.

As a consequence of the specific requirements on the object recognition module, and of the analysis conducted in Chapter 7, the current experimental implementation relies on the AlexNet architecture [113]. The network weights are initialized on the task of object classification from ImageNet [113], with the final fully-connected layer re-instantiated in order to match the different cardinality and task. Before feeding the image to the network, the value 0.5 is subtracted as an approximation of the mean-image subtraction technique that allows to speed up training and fine-tuning of the neural model.

### 9.2.2 Illuminant Estimation network

The Illuminant Estimation network takes an input RGB image and predicts the scene illuminant. For the experiments described here, the illuminant model has been limited to a diagonal matrix, i.e.  $\rho^e = (\rho_R^e, \rho_G^e, \rho_B^e)^T$ , although alternative combinations are

possible [22]:

$$\begin{bmatrix} R_{\text{out}} \\ G_{\text{out}} \\ B_{\text{out}} \end{bmatrix} = \begin{bmatrix} 1/\rho_R^e & 0 & 0 \\ 0 & 1/\rho_G^e & 0 \\ 0 & 0 & 1/\rho_B^e \end{bmatrix} \cdot \begin{bmatrix} R_{\text{in}} \\ G_{\text{in}} \\ B_{\text{in}} \end{bmatrix} \quad (9.1)$$

This representation also matches the annotation associated to most datasets for color constancy, which provide ground truth illuminants in terms of triplets [178, 53, 43].

During the end-to-end training of the whole IEOR network, input data is artificially augmented by applying a random illuminant extracted from a Gaussian distribution with mean 1 and standard deviation 0.3. Once training is completed as described in Section 9.2.1, IE can be used as a standalone network for color correction on any image, which is not necessarily depicting the classes seen during the training phase.

The IE module is also based on AlexNet, which was shown in the past to perform well on color constancy [135], and images are preprocessed by subtracting 0.5 as with the OR module. The color-corrected image obtained by applying the estimated illuminant to the input image is clipped between 0 and 1. This operation is necessary to avoid feeding to the pre-trained OR values in a range that was never encountered during training, and is also recreating what would happen by displaying the color-corrected image.

## 9.3 Experiments

These experiments are designed to evaluate how the proposed illuminant estimation network, which does not require any illuminant ground truth, compares to other illuminant estimation methods on standard datasets. Two possible normalization techniques are considered (global and channel-wise), and results are compared with state of the art methods.

### 9.3.1 Experimental setup

The training phase requires a classification task where color is a discriminative feature, in order to properly drive the learning process. To this extent, the vegetables subset from the **VegFru** dataset [100] fits the desired criteria, as shown in Figure 9.2. It contains more than 90000 images belonging to 200 vegetable classes. At test time only the IE module is applied (namely, the branch related to object recognition is not used) and the proposed solution is evaluated on two widely-adopted benchmarks for color constancy: the Shi-Gehler [178] and the more recent dataset from National University of Singapore (NUS) [43]. **Shi-Gehler** [178] is a linear reprocessing of the original RAW images from



Fig. 9.2 Samples from VegFru dataset, showing the importance of color in vegetable class discrimination

the Gehler dataset [79]. It contains 568 images of different scenarios with a ColorChecker in each picture as a support for the ground truth extraction. At test time, the area corresponding to the ColorChecker is edited out to prevent the illuminant estimation algorithm from directly using it. Results on the Shi-Gehler datasets are computed with the ground truth from [23]. NUS from National University of Singapore [43] contains a total of 1853 images coming from 9 digital cameras of various brands. The type of content is similar to Shi-Gehler, with both indoor and outdoor scenes, and ground truth extracted from a ColorChecker. A unique characteristic of this dataset is the presence of the same scenes taken from different cameras. Results on NUS are reported as the average performance on each camera.

Evaluation is based on the well-established angular recovery error [81, 43], which compares the estimated ( $\rho^e$ ) and reference ( $\rho^r$ ) illuminant triplets regardless of their magnitude:

$$err_{ang} = \arccos \left( \frac{\rho^{eT} \rho^r}{\|\rho^e\| \|\rho^r\|} \right) \quad (9.2)$$

### 9.3.2 Input normalization

During training, a dataset annotated for classification is used, as the model learns to minimize the cross-entropy error on a given classification task. At test time, instead, it is desirable to evaluate the proposed solution on standard datasets for color constancy, which are typically provided with only the ground truth illuminant. In order to account for the

### 9.3 Experiments

Table 9.1 Performance of different algorithms for illuminant estimation. The proposed solution (in its global- and channel-normalization variants) can be directly compared with cross-dataset learned methods. Values reported for parametric solutions [43] and [72] are selected with the best configuration of test parameters. Learned methods with in-dataset training are also reported, although a direct comparison cannot be performed.

| Method                         |                        | Angular error (Shi-Gehler [178]) |        |       |        | Angular error (NUS [43]) |        |       |        |
|--------------------------------|------------------------|----------------------------------|--------|-------|--------|--------------------------|--------|-------|--------|
|                                |                        | Mean                             | Median | Std   | Max    | Mean                     | Median | Std   | Max    |
| Baselines                      | Unchanged              | 13.62°                           | 13.55° | 2.85° | 27.37° | 19.50°                   | 18.82° | 1.90° | 25.83° |
|                                | Greyworld              | 7.35°                            | 6.70°  | 3.78° | 25.84° | 4.59°                    | 3.64°  | 3.57° | 22.61° |
|                                | Regression             | 5.96°                            | 5.31°  | 3.47° | 19.88° | 5.19°                    | 3.90°  | 4.16° | 22.07° |
| Param.                         | Akbarinia et al. [3]   | 3.8 °                            | 2.4 °  | -     | -      | -                        | -      | -     | -      |
|                                | Cheng et al. [43]      | 3.52°                            | 2.14°  | -     | 28.35° | 3.02°                    | 2.12°  | -     | 17.24° |
|                                | Funt et al. [72]       | 3.2 °                            | 2.3 °  | -     | 21.7 ° | -                        | -      | -     | -      |
| Learned<br>(cross-<br>dataset) | <b>This work (gl.)</b> | 4.84°                            | 4.12°  | 3.22° | 20.80° | 4.88°                    | 4.17°  | 3.11° | 18.70° |
|                                | <b>This work (ch.)</b> | 5.48°                            | 4.81°  | 3.21° | 19.88° | 4.32°                    | 3.37°  | 3.56° | 22.36° |
|                                | Joze et al. [108]      | 6.5 °                            | 5.1 °  | -     | -      | -                        | -      | -     | -      |
|                                | Gao et al. [75]        | 5.03°                            | 3.39°  | -     | -      | -                        | -      | -     | -      |
|                                | Lou et al. [135]       | 4.7 °                            | 3.3 °  | 5.3 ° | -      | -                        | -      | -     | -      |
| Learned<br>(in-<br>dataset)    | Chakrabarti [37]       | 3.59°                            | 2.96°  | -     | 21.58° | 3.04°                    | 2.40°  | -     | 15.38° |
|                                | Bianco et al. [23]     | 2.63°                            | 1.98°  | -     | 14.77° | -                        | -      | -     | -      |
|                                | Oh et al. [148]        | 2.16°                            | 1.47°  | -     | -      | 2.41°                    | 2.15°  | -     | -      |

potential discrepancies between training and test images, several techniques for input normalization are adopted. Gamma correction is applied whenever a linear dataset is used at test time, since the employed training classification datasets are already processed for gamma. The estimated illuminant is then re-corrected before the final evaluation with the provided ground truth. Then, two alternative techniques were designed and integrated, in order to ensure the range of image values is stable. These techniques are based on the generation of a support diagonal illuminant:

1. Global normalization, which brings the overall average of the input image to a fixed value without changing the relationship between single channels.
2. Channel normalization, that modifies each channel independently, thus affecting the original illuminant.

The complete processing pipeline is shown in Figure 9.3.

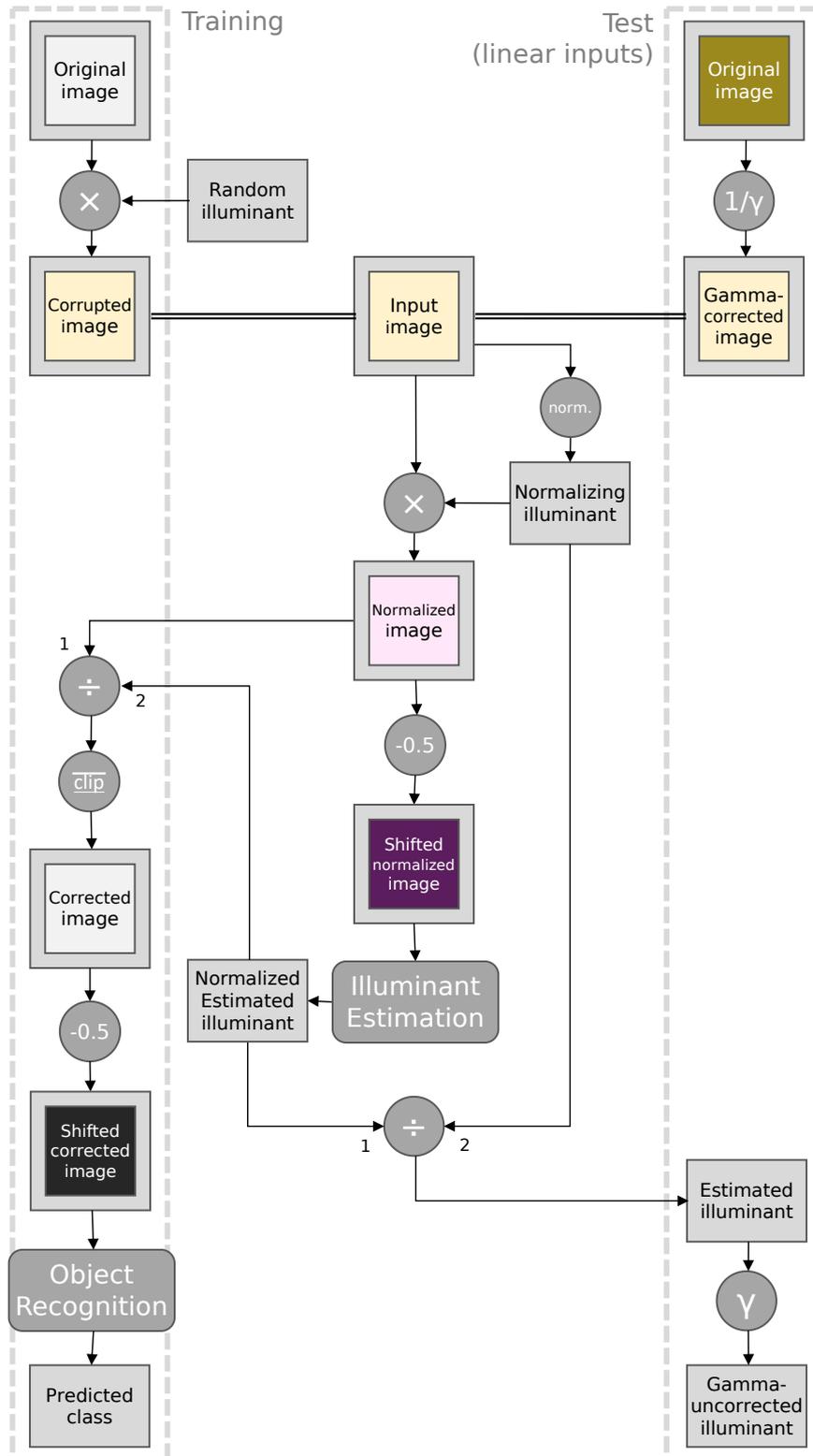


Fig. 9.3 Complete pre- and post-processing for the proposed strategy for illuminant estimation, including gamma correction, input normalization, and mean-value subtraction.

### 9.3.3 Results

Table 9.1 reports the results of the two variants of the proposed method on Shi-Gehler and NUS datasets, compared to different baselines and different algorithms from the state of the art.

Global normalization brings to very similar results on the two datasets, providing a robust performance assessment of the proposed solution. Channel normalization leads instead to contrasting conclusions on each dataset, suggesting a more challenging nature of NUS over Shi-Gehler. As channel normalization destroys the relationship between channels, in fact, it essentially discards any potentially misleading information about the original illuminant. In the case of NUS, the proposed model benefits from this preprocessing, as it is forced to estimate the proper illuminant directly from the image content.

In order to assess the effective advantage that comes from using a classification-based loss, the Illuminant Estimation module is also trained for direct regression (third row) over the color-augmented examples of VegFru, i.e. training is performed without OR. Performance on both sets shows that the classification-based strategy can lead up to a 19% improvement in angular error, thus highlighting the relevance of the proposed approach.

The “Learned (cross-dataset)” block includes data-driven models that are trained on one dataset and tested on a different one, instead of adopting a more traditional training-test split of the same dataset. The proposed solution was trained on the VegFru dataset, and is therefore best compared with methods from this category, which are either trained on the GrayBall dataset [108, 135] or on SFU-Lab [75]. The IEOR network described in this chapter outperforms the solutions by Joze et al. [108] and Gao et al. [75], while producing comparable results to Lou et al. [135]. For completeness, other state-of-the-art methods are also included, trained on the training-set portion of the same dataset used for evaluation (“Learned (in-dataset)”), although direct comparison with these solutions is not applicable. Parametric methods [43] and [72] present different results with varying parameters and configurations. For such solutions, only the best configuration is reported, which was directly selected from the test set performance.

Figure 9.4 shows some examples predictions with the corresponding angular error, in order to provide a visual guide for the interpretation of the reported performance values.

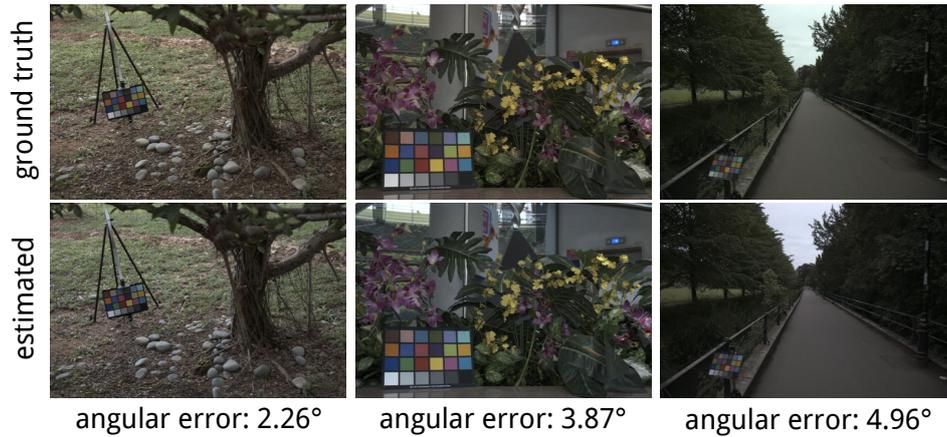


Fig. 9.4 Example color corrections of the proposed method at different levels of angular error.

### 9.3.4 Analysis on the training-set color bias

The illuminant distribution present in the classification training dataset plays an important role in the proposed method.

IE implicitly learns to replicate the average illuminant of the training set. If this average is not neutral, the predicted illuminants on the test set will not, in general, match the provided ground truth. By using an oracle to shift the predictions around the average ground truth illuminant, it is possible to further reduce the angular error on Shi-Gehler from  $4.84^\circ$  to  $4.60^\circ$  (5%), and from  $4.32^\circ$  to  $4.23^\circ$  on NUS (2%).

Secondly, a high variability in the illuminant during the initial pre-training of OR may make it excessively robust to color variations, thus compromising the consequent training of IE. This condition is particularly hard to avoid: since dataset heterogeneity typically results in better classification performance [16], dataset curators tend to implicitly or explicitly encourage a certain degree of illuminant variability. To this extent, a possible direction for improvement is to include unsupervised white balancing, as a preprocessing step for the classification-oriented training set itself.

# Chapter 10

## Distance estimation

*“Even though you’re so close to me, you’re still so distant, and I can’t bring you back”*

Linkin Park - With You

Distance estimation is a topic of particular relevance due to its various applications, which include autonomous and semi-autonomous driving [197], analysis of video surveillance cameras for traffic safety analysis [13] or information forensics [64], and multimedia processing for artistic purposes [166]. Further applications can be found specifically for monocular distance estimation, in fields where the imaging device should be as compact as possible (e.g. endoscopy and laparoscopy scans [96]), or whenever it is preferable to introduce depth vision without an expensive hardware upgrade, as with biomedical and astronomical technologies [85].

Several existing works address this topic by ignoring the perspective geometry of image formation. As such, they require a re-calibration or even re-training phase for each specific hardware setup. This chapter presents a universal representation that makes it possible to jointly use data coming from different sensors, and thus to generate a single, more powerful, hardware-agnostic model. The case study of street and urban environments from Chapter 4 is here taken once again into consideration, thus forming a natural extension of semantic segmentation.

The depth perception of the human visual system exploits different cues:

- Binocular cues combine the signals received from both eyes in order to reconstruct a 3-dimensional geometry. They are the source of inspiration for stereopsis-based systems such as the Microsoft Kinect [212].
- Motion cues exploit the time-varying signals received from each eye. The same technique is also used in depth-from-motion methods [160].

- 
- Muscular cues integrate information coming from the voluntary or involuntary movement of eye parts [90, 185].
  - Monocular cues use only the retinal signal of a single eye. Related automatic methods may explicitly rely on specific elements such as texture, shading and defocus [158, 65], introduce additional cues such as structured light patterns [174], or implicitly extract all the necessary features from the input image [190, 84, 114].

Binocular and motion cues provide the most reliable source for automatic distance estimation, as they are based on a rigorous geometric model [101]. Monocular cues are generally less robust [139] but still present several advantages. For instance, they allow distance estimation in absolute terms provided that the subject size is known [188], and their implementation is inherently cheaper than that of binocular cues, as it requires a single imaging device. Since monocular and binocular methods exploit two different kinds of image features, distance estimation can then be made more robust by integrating such complementary and decorrelated methods.

Taking inspiration from the monocular family of visual cues, the work presented in this chapter is focused on the concept of “familiar size” [94]: during the development of the human visual system in our first months of life, we get to implicitly learn the relationship between the apparent size of a known object on our retina, and its actual distance from us [207]. The same technique is further exploited by hikers and building surveyors, with the so-called *rule of thumb*, where the subject’s apparent size is measured relative to an object at arm distance (*e.g.* the measurer’s own thumb). In these cases, knowing the subject size in real life is fundamental to actually perform an estimation of its distance, hence the term “familiar size”. The necessity for semantic awareness is fulfilled, in the following, through the use of semantic segmentation networks such as the model described in Chapter 4 and other related architectures. In this chapter, a unified representation of distance is proposed, taking inspiration from the familiar size cue. This modelization is independent of device-specific characteristics, and as such it allows the integration of information from different sensors to train richer data-driven models, which take advantage from large and heterogeneous datasets to improve prediction [16]. Models trained with such technique can then be applied to data generated by different devices. This process could also bring to common representation acquisitions performed with different sensors on the same scene, in order to combine the different sources of information and produce a more robust measurement.

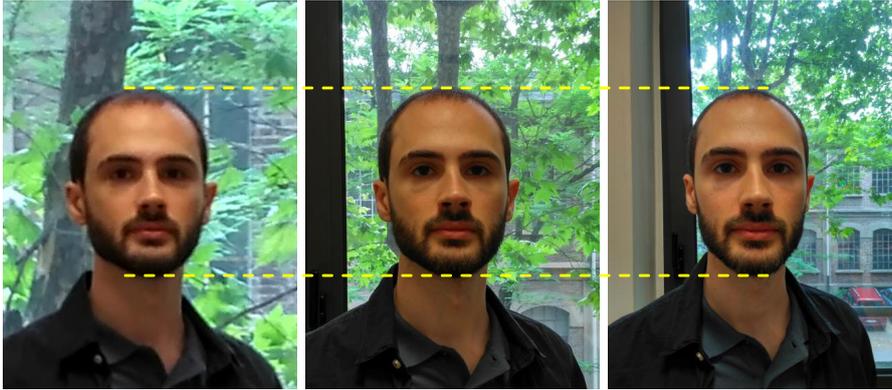


Fig. 10.1 For all three pictures, the subject size (face) is the same both in real life and in picture. Knowing the camera parameters is essential in determining different distances of the face from the camera.

## 10.1 Background and related works

The depth perception exploited by human vision follows the pinhole model of image formation, which can be formalized and applied to digital images as follows:

Let  $F$  be a person-specific or device-specific parameter (px).

Let  $real\_size$  be the known subject size<sup>1</sup> in real life (m).

Let  $apparent\_size$  be the measured subject size (px):

$$distance = \frac{real\_size}{apparent\_size} F \quad (10.1)$$

In the case of digital acquisition devices, parameter  $F$  corresponds to the camera focal length expressed in pixels, which in turn can be computed from the focal length in meters as:

$$F = \frac{image\_size}{sensor\_size} F_{metric} \quad (10.2)$$

The distance value associated to a specific subject therefore depends on hardware-specific elements ( $sensor\_size$ ) as well as variable camera configuration (i.e. zooming through  $F_{metric}$ ). This is visually shown in Figure 10.1.

Several works [95, 60] are based on the perspective geometry of Equation (10.1) to perform distance-related reasoning. This formulation requires an explicit localization of each known subject in the image in order to effectively measure their apparent size. Such approach presents its limitations when dealing with severe occlusions and unconventional

<sup>1</sup>All sizes refer to linear size and not surface size, so they can be either height or width. The rest of this chapter will always refer to width measures for all experiments.

poses. In the case of human subjects, for example, *real\_size* is their (known) height in meters, and *apparent\_size* is their (unknown) height in pixels. If the person is standing straight, *apparent\_size* is simply the height of the bounding box resulting from a person detector. If the subject is in any other pose, instead, it is necessary to follow their body structure in order to accurately measure their height in pixels.

An alternative solution is to directly predict the distance associated with each pixel, completely ignoring the image formation model. Such formulation is typically implemented with Convolutional Neural Networks (CNN), and has the advantage of addressing different problems with a unique solution. It can, in fact, handle different poses and cluttered scenes indiscriminately [133]. It is not bound to specific object classes, and it is able to implicitly handle fine-grained models (*e.g.* a child’s average height is different from that of an adult). Several recent works adopt this dense formulation of the problem. Uhrig et al. [190] presented a multi-objective CNN that simultaneously predicts objects class, discretized distance, and a novel representation used to separate multiple instances. Experiments were performed on the CityScapes [56] and KITTI [80] datasets separately. As highlighted in Equation (10.1), in fact, the same elements can be associated with different distance values when dealing with different acquisition devices, due to the effect of focal length  $F$ . The KITTI Vision Benchmark [80] encourages the development of models predicting stereo-pair disparity in place of distance, by providing ground truth annotations in such format. This formulation, however, is also tied to a specific hardware setup, as it depends on the baseline distance between the two cameras used. Godard et al. [84] introduced a learning loss based on predicted disparity that optimizes the consistency between left-right pairs used in the training phase. They also observed how training on data from different devices actually deteriorates the performance of their solution, again due the different camera calibrations. Ladicky et al. [114] reformulated the problem of distance estimation as predicting the probability of each pixel belonging to a so-called “canonical depth”. This interpretation reduces the dependency from device-specific parameters, and was proven effective when training on multiple datasets. Aiming at a similar goal, the following sections present an alternative representation that is independent of sensor configuration, showing its advantage in improving the performance of two neural network-based models. Eigen et al. [63] approached the problem of distance estimation with a solution based on Conditional Random Fields (CRF), and introduced a training loss that is only partially scale-invariant, in order to focus on relative depth details without compromising overall accuracy. Other works also adopted CRF-based approaches to the problem [119], to induce a local reciprocal influence to the final estimation. Taking inspiration from this strategy, a Fully Convolutional Neural Network architecture (FCN)

is here adopted for experimental evaluation of the proposed representation, as it allows for a dense (pixel-precise) prediction based on a local analysis [127].

## 10.2 Proposed representation for distance information

In order to integrate data coming from two different devices  $A$  and  $B$  there are two possible solutions:

1. Convert the training distance information from  $A$  and  $B$  into real-life (metric) size, as a consequence of Equation (10.1):

$$real\_size = distance_{\{A,B\}} \cdot \frac{apparent\_size_{\{A,B\}}}{F_{\{A,B\}}} \quad (10.3)$$

(The fact that Equation (10.3) can be applied to either device is denoted by  $\{A, B\}$ ). The predicted size will then be converted back to distance, based on the desired camera parameters from either  $A$  or  $B$ .

2. Transform all training data from  $A$  according to a reference set of camera parameters from  $B$ , by combining Equation (10.3) with Equation (10.1):

$$distance_B = distance_A \cdot \frac{F_B}{F_A} \cdot \frac{apparent\_size_A}{apparent\_size_B} \quad (10.4)$$

The predicted distance will always be based on parameters from  $B$ , but it can later be transformed back to any desired setting with the same formulation.

The two alternatives are tightly related, as Equation (10.4) is a direct extension of Equation (10.3). The second one is an explicit prediction of the distance information, inherently depending on camera parameters. The first one is directly related to image content, and it is totally uncorrelated to any specific sensor configuration. This is the formulation that will be used in the following, applying the advantages of dense prediction models to the pinhole camera model. Such approach allows the gathering of training data from different sources, increasing both cardinality and diversity in the dataset.

A practical example of the proposed representation is depicted in Figure 10.2: each pixel is described by the size, in real life, of what it portrays. When the subject is further away, each pixel will span a larger area, thus *real\_size* will be larger. This concept is straightforward for flat surfaces that are parallel to the acquisition device plane. In all

## 10.2 Proposed representation for distance information

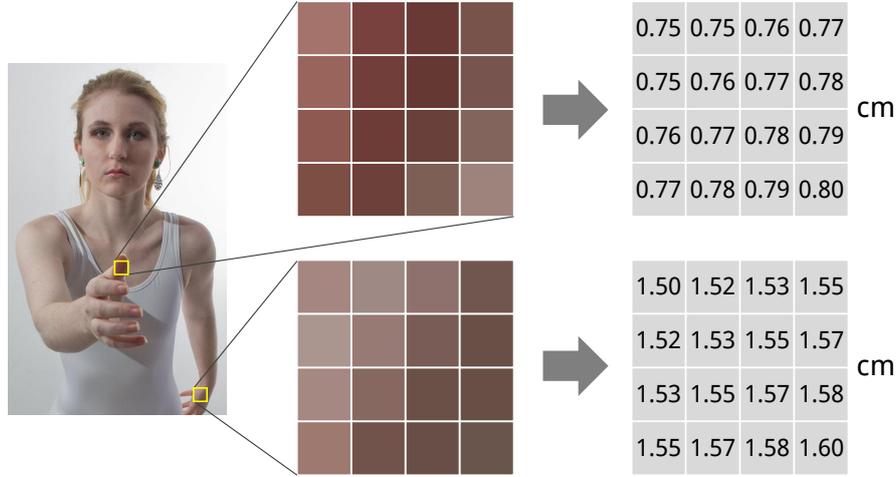


Fig. 10.2 Visualization of the proposed size-based representation of distance: each pixel is associated with the size of what it represents in real life. This is obtained by combining distance information with the camera parameters. Image credit Courtney Simonds.

other cases, it is generalized to the average size of what is depicted by the pixel itself, or:

$$average\_size = \frac{\int_{x_i}^{x_{i+1}} \int_{y_j}^{y_{j+1}} real\_size \, dx \, dy}{(x_{i+1} - x_i) \cdot (y_{j+1} - y_j)} \quad (10.5)$$

where  $x$  and  $y$  refer to a world coordinate system aligned to the acquisition sensor’s coordinate system, while  $x_i$  and  $y_j$  are used to delimit the frustum behind each pixel. For legibility, the average size will be also referred to as *real\_size*.

An additional advantage of performing a prediction with pixel-precision is that Equation (10.1) is no longer dependent on *apparent\_size*, which is reduced to one. The eventual conversion from *real\_size* to *distance* is thus simplified to a multiplication by the camera focal length expressed in pixels ( $F$ ). Typically, the methods based on the pinhole camera model, such as those described in Section 10.1, use the opposite approach to the same formulation, as they consider the subject *real\_size* to be known, and predict its *apparent\_size* via object detection.

The problem formulation proposed here is based on the “familiar size” monocular cue for distance estimation. For this reason, it is necessary to constrain both training and prediction to those image regions that correspond to a known object class. In particular, the experiments presented in the following will be initially focused on the “people” class. This decision is guided by two factors: the availability of training data, as highlighted in Section 10.3, and the high relevance of human subjects in personal photo collections

## 10.2 Proposed representation for distance information

---

[142]. Despite the particular setup employed here, though, the method can be effortlessly extended to other object classes, as shown with further experiments.

Prediction of subject size is inherently tied to the semantics of image content. It is, in fact, fundamental to understand the nature of the portrayed elements, in order to estimate their size. For this reason, a Fully Convolutional Network (FCN) will be used, internally representing the semantics of the scene by using layers previously trained on object recognition [133, 181]. The activation of layer *conv5-3* from a VGG-19 net [181] is processed with two “Conv-ReLU-DropOut” blocks, followed by a “Conv” block. The result is upsampled using a convolution-transpose layer, summed to the activation of layer *pool4*, upsampled again, summed to activation of layer *pool3* and upsampled one last time. When properly trained, this finally produces an input-sized prediction corresponding to the content size in real life of each pixel (*i.e.* *real\_size* in Equation (10.1)). A similar approach was used in Chapter 3 to highlight the salient regions of a picture, exploiting its semantic content without explicitly defining any specific object class.

Training a FCN aims to minimize the average error between prediction  $P$  and ground truth  $GT$  computed at each position  $i$  in image coordinates  $I$ :

$$L = \frac{\sum_{i \in I} \text{error}(GT_i, P_i)}{|I|} \quad (10.6)$$

Here  $|I|$  identifies the cardinality of set  $I$ . In the experimental setup described later, *error* is implemented as a cross entropy loss on discretized levels.

The ground truth contains *real\_size* information, which is typically quite sparse due to the nature of depth-acquisition devices. This reduces the set of pixels effectively useful for training. As only human subjects will be taken in consideration, then, the training data is even sparser, as shown in Figure 10.3.

To take into account this particular condition, a mask pointing only to relevant pixels is created, and the annotation “holes” are filled using a nearest neighbor approach. This strategy prevents any artifact during future manipulations of the ground truth data, such as image resizing. The relevance *mask* is then introduced in the loss computation multiplying it by the pixel-wise error, and the result on all relevant pixels  $M$  is averaged:

$$L' = \frac{\sum_{i \in M} \text{error}(GT_i, P_i) \cdot \text{mask}_i}{|M|} \quad (10.7)$$

Thanks to this operation, the backpropagation algorithm excludes any mismatched prediction on non-relevant pixels, updating the gradients only as a function of pixels

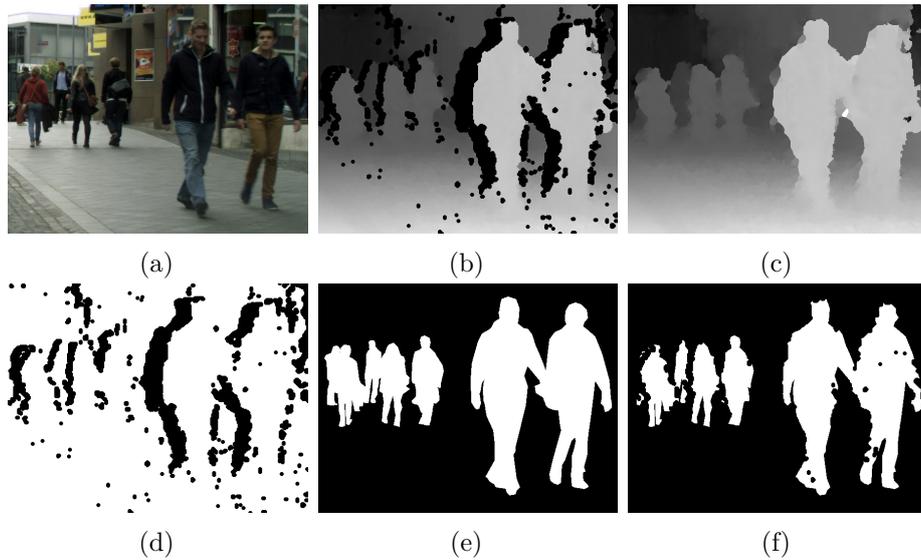


Fig. 10.3 Mask generation and ground truth preprocessing. a) The RGB input image. b) The corresponding ground truth. c) The ground truth after hole-filling. d) The initial mask from invalid ground truth values. e) The person-specific mask. f) The final relevance mask.

indicated by the mask. The final architecture is shown in Figure 10.4. More details on how to produce and process the human-subject masks are given in Section 10.3.

## 10.3 Datasets transformation to common representation

Four different datasets were used for the experiments: CityScapes [56], SYNTHIA [170], KITTI [80], and RGB-D People [183]. Sample images are shown in Figure 10.5. Each dataset is originally encoded using different conventions and formats, due to the different sensors involved in their acquisition. Thus each required a specific preprocessing, described in the following, aimed at obtaining the proposed common representation *real\_size*. Eventually, dataset-independent processing such as data augmentation was also performed as described in Section 10.4.1.

The **CityScapes** dataset [56] (CSC), used for semantic segmentation in Chapter 4, contains stereo video sequences from European city streets. It contains 5000 high resolution stereo pairs ( $2048 \times 1024$ ) divided into 2975 training images, 500 validation images, and 1525 test images. The attached corresponding disparity maps were computed using SemiGlobal Matching [93]. Conversion from *disparity* to *real\_size* was performed

### 10.3 Datasets transformation to common representation

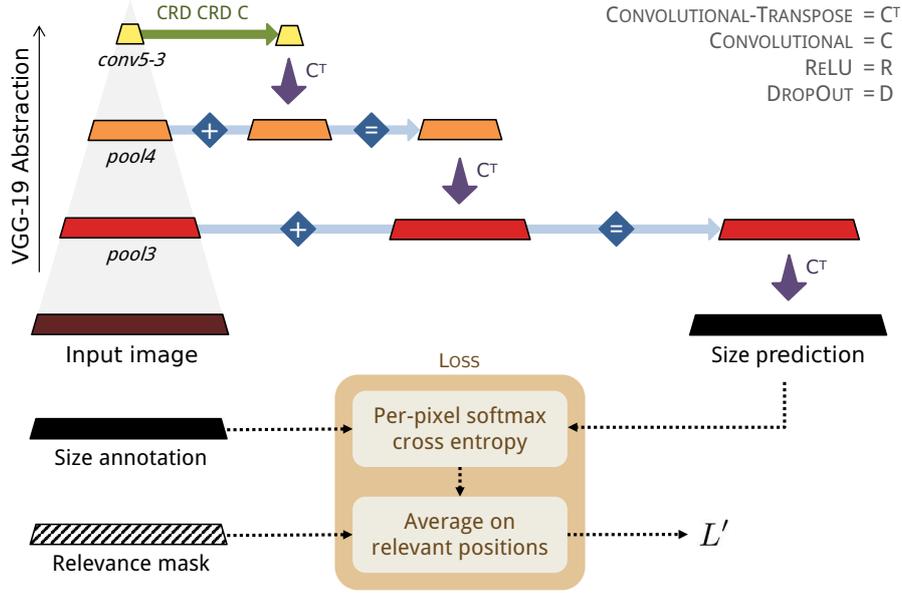


Fig. 10.4 Schematic view of the Fully Convolutional Network employed for distance estimation through size prediction. Intermediate activations of a VGG-based processing [181] are resized and combined in order to implement a multi-resolution analysis. The final prediction is evaluated only on relevant image positions

with the following simplification on the formulas for disparity-to-distance and distance-to-size:

$$real\_size = \frac{distance}{F} = \frac{F \frac{baseline}{disparity}}{F} = \frac{baseline}{disparity} \quad (10.8)$$

Fine class annotations are provided for the training and validation sets, including the “person” and “rider” classes, which were used as training masks for the “people”-specific experiments. Further annotations used in later experiments are “car”, “bike” and “motorbike”.

The **SYNTHIA** dataset [170] (SYN) is a collection of  $1280 \times 760$  synthetic frames rendered from virtual cities in different weather and lighting conditions. For the purpose of this work, only the subset called SYNTHIA-RAND-CITYSCAPES was used, which originally contains 9400 instances, sampling one every fourth image in order to balance the different training sets. The distance information is stored as 16-bit encoded images, representing the distance value for each pixel in centimeters. Conversion to metric size in real life was thus obtained simply dividing by 100 and by the focal length of the virtual camera. The semantic classes of this dataset are annotated with criteria similar to those used in CityScapes. In this case, though, the information was retrieved directly from the scene 3D model instead of relying on manual annotation.

### 10.3 Datasets transformation to common representation

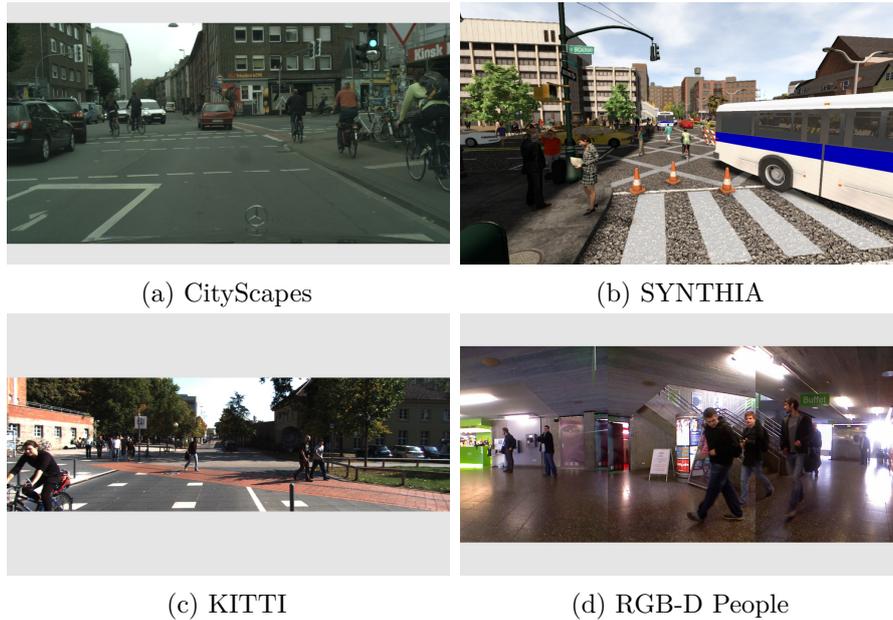


Fig. 10.5 Sample images from each dataset show the difference in content and format among the used training data.

The **KITTI** Vision Benchmark Suite [80] (KIT) provides multi-sensor recordings acquired in and around the city of Karlsruhe, Germany. The “raw data” section includes rectified images and LiDAR distance points. The “Campus” subset and sequence 72 of the “City” subset were used, due to the high presence of human subjects. This results in a total of 2308 images, with varying resolution under  $1392 \times 512$  pixels. LiDAR-acquired 3D points were first projected to the image plane, using the provided projection matrix. This produced an extremely sparse set of distance measurements on image coordinates, which were dilated using a  $3 \times 3$  diamond-shaped structuring element. Conversion to *real\_size* was then again obtained dividing the distance by the provided focal length  $F$ , following Equation (10.1). This dataset offers no pixel-precise annotation about the location of people and other classes of interest. Such information can, however, be automatically generated with any of the semantic segmentation networks described in Chapter 4. Specifically, the FCN model described by Long et al. [133] was here used for architectural continuity, and the corresponding semantic classes of interest were selected to constrain the training and evaluation phases to relevant pixels only.

The **RGB-D People** Dataset [183] (RDP) contains 3399 images and disparity acquisitions from three vertically-mounted Kinect sensors placed in a university hall. For the experiments involved in this work, only two of the three devices were used, for data balancing reasons. The resolution of the RGB images is  $480 \times 640$  pixels. The disparity

| Abbr.  | Dataset name | Subset             | Total | Used        |
|--------|--------------|--------------------|-------|-------------|
| CSC    | CityScapes   | Training set       | 2975  | <b>2498</b> |
| SYN    | SYNTHIA      | RAND-CITYSCAPES    | 9400  | <b>2350</b> |
| KIT    | KITTI        | Campus and City-72 | 2308  | <b>2294</b> |
| RDP    | RGB-D People | Sensors 0 and 1    | 2266  | <b>1937</b> |
| CSCval | CityScapes   | Validation set     | 500   | <b>441</b>  |

Table 10.1 Cardinality of the chosen datasets. The last column reports the number of images effectively used, after sampling and removing images without human subjects

data coming from the Kinects was first converted into metric distance using the following formula from Spinello et al. [183]:

$$distance = \frac{F \cdot baseline \cdot 8}{Vmax - disparity} \quad (10.9)$$

$Vmax$  is the maximum measurable value, and  $baseline$  is the distance between Infrared (IR) projector and IR camera. Radial and tangential distortions were then corrected for both depth and color data using the respective intrinsic matrices. This was done to align the two sources of data. The distance points were brought to IR-projector-world coordinates, moved to RGB-camera-world coordinates, and eventually reprojected to the image plane. The distance data, now registered to the color data, was finally converted to size in real life by dividing it by the camera focal length. This dataset does not provide any pixel-precise segmentation masks, which were therefore generated once again using the method proposed by Long et al. [133].

The processed datasets were sampled with the general intention of balancing the different data, and further reduced by excluding images without any subject belonging to the class of interest. The final cardinalities for the “people” class are shown in Table 10.1. Thanks to the achieved common representation, these datasets can be effortlessly joined in various combinations: Section 10.4 shows the effect of training on different subsets of the collected dataset.

## 10.4 Experiments

Experiments are structured towards a quantitative evaluation of the proposed representation. To this end, the adopted neural network was trained on human subjects from different subsets of the datasets shown in Table 10.1, and tested every time on a fixed

set: the validation set of CityScapes [56], which is composed of 500 images annotated with both distance information and pixel-precise semantic classes. The goal here is to test whether building a larger and more diverse training set, which is made possible by the proposed representation for unifying distance information, can in fact improve performance on a given test set. This representation is, however, general-purpose, and as such it was further tested on additional semantic classes, such as bikes, motorbikes, and cars, and its benefits evaluated on another recent method for depth estimation [147].

Preliminary tests were also conducted in order to select various hyper-parameters. All the experiments were consequently run with logarithmic (base 10) quantization of *real\_size* values in 100 classes between 0 and 0.1. The current experimental setup, in fact, is essentially transforming a regression problem into a classification problem, as already proposed by Uhrig et al. [190]. Finally, following a preliminary selection phase, mini-batches composed of 15 images were adopted, and the learning rate was set to  $10^{-4}$ .

### 10.4.1 Training preprocessing

During training, online data augmentation was applied coherently with Chapter 3, including random horizontal flip, random gamma correction between 0.3 and  $0.3^{-1}$ , and random cropping with varying size. All crops were then rescaled to a fixed size, set to  $256 \times 256$  pixels, in order to exploit minibatch parallel computation during the network training. After resizing the crop, the annotation values for *real\_size* were adjusted accordingly, multiplying them by the ratio between the original crop size and the final dimensions after resizing. Each training dataset presents a different distribution of *real\_size* values, and since the resizing operation affects such distribution, it can be used as a tool to make one dataset more similar to the target one used in the test phase. For all datasets, preliminary tests with three different cropping size ranges were performed:  $64 \div 256$  to make the objects bigger and therefore move the distributions to lower values,  $256 \div 256$  to keep the distributions unchanged as there will be no resizing involved, or  $256 \div 640$  to make objects smaller and move the distributions to higher values. The final settings for each dataset are shown in Figure 10.6. Given the high sparsity of relevant ground truth data, as highlighted in Section 10.2 and Figure 10.3, steps have been taken to verify that the random crops always include a minimum amount of pixels belonging to the class of interest (i.e. human subjects in the first set of experiments).

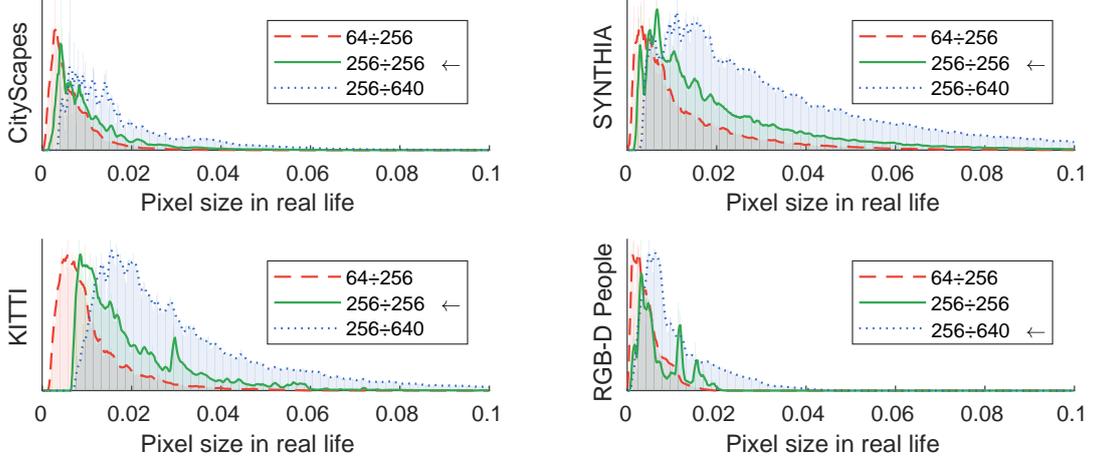


Fig. 10.6 Distribution of *real\_size* values of the four datasets, as impacted by three different crop size ranges. Selected ranges are highlighted inside the legend

### 10.4.2 Evaluation procedure

In order to evaluate the models, the classification prediction was de-quantized using the central values of the discretization levels applied during preprocessing. Predicted ( $P$ ) and ground truth ( $GT$ ) *real\_size* data was then transformed back to *distance* values through multiplication by the proper focal length  $F$ . The values were finally compared with the ground truth using Mean Absolute Error (MAE) as well as several error measures from Eigen et al. [63], including Absolute Relative Difference (ARD), Squared Relative Difference (SRD), linear and logarithmic Root Mean Square Error (RMSE):

$$\text{MAE} = \sqrt{\frac{\sum_{i \in M} \|P_i \cdot F - GT_i \cdot F\|}{|M|}} \quad (10.10)$$

$$\text{ARD} = \sqrt{\frac{\sum_{i \in M} \|P_i \cdot F - GT_i \cdot F\| / GT_i \cdot F}{|M|}} \quad (10.11)$$

$$\text{SRD} = \sqrt{\frac{\sum_{i \in M} \|P_i \cdot F - GT_i \cdot F\|^2 / GT_i \cdot F}{|M|}} \quad (10.12)$$

$$\text{RMSE}_{\text{lin}} = \sqrt{\frac{\sum_{i \in M} (P_i \cdot F - GT_i \cdot F)^2}{|M|}} \quad (10.13)$$

$$\text{RMSE}_{\text{log}} = \sqrt{\frac{\sum_{i \in M} (\log(P_i \cdot F) - \log(GT_i \cdot F))^2}{|M|}} \quad (10.14)$$

|    | Training datasets |              |             |              | Error measures on CityScapes validation set |              |             |                     |                     | Rank  |          |
|----|-------------------|--------------|-------------|--------------|---|--------------|-------------|---------------------|---------------------|---|----------|
|    | CSC<br>[56]       | SYN<br>[170] | KIT<br>[80] | RDP<br>[183] | MAE   | ARD          | SRD         | RMSE <sub>lin</sub> | RMSE <sub>log</sub> | RMSE <sub>log</sub>   |          |
| 1  | ✓                 |              |             |              | 22.51                                       | 0.524        | 16.07       | 30.58               | 0.677               |  | 8        |
| 2  |                   | ✓            |             |              | 46.14                                       | 1.980        | 210.4       | 69.17               | 1.055               |  | 10       |
| 3  |                   |              | ✓           |              | 15.10                                       | 0.458        | 12.36       | 23.51               | 0.473               |  | 4        |
| 4  |                   |              |             | ✓            | 21.05                                       | 0.429        | 14.45       | 29.78               | 0.650               |  | 6        |
| 5  | ✓✓                |              |             |              | 22.70                                       | 0.521        | 16.20       | 30.83               | 0.690               |  | 9        |
| 6  | ✓                 | ✓            |             |              | 22.03                                       | 0.643        | 17.95       | 29.53               | 0.658               |  | 7        |
| 7  | ✓                 |              | ✓           |              | <b>8.48</b>                                 | <b>0.177</b> | <b>4.33</b> | <b>16.69</b>        | <b>0.255</b>        |  | <b>1</b> |
| 8  | ✓                 |              |             | ✓            | 9.93  | 0.182        | 6.88        | 19.62               | 0.270               |  | 2        |
| 9  | ✓                 |              | ✓           | ✓            | 9.19  | 0.183        | 4.68        | 17.64               | 0.279               |  | 3        |
| 11 | ✓                 | ✓            | ✓           | ✓            | 20.24                                       | 0.666        | 19.62       | 27.94               | 0.590               |  | 5        |

Table 10.2 Error measures on CityScapes validation set (average distance 40.39m) for the “people” class with different training subsets, collected and transformed according to the proposed representation. For all considered measures, a lower value is better.

Note that the logarithmic transformation in Equation (10.14) actually removes the contribution of any coefficient equally applied to annotation and prediction, including the effect of focal length  $F$ . Here  $||x||$  denotes the absolute value, and  $|M|$  the cardinality of set  $M$ . Equations (10.10) to (10.14) were computed only over pixels highlighted by the relevance mask  $M$ , as done with the loss in Equation (10.7). The final errors were obtained averaging the results from all validation images.

All experiments were run for 10000 iterations, and the best performing iteration on the validation set according to logarithmic RMSE was eventually selected.

### 10.4.3 Experimental results

These experiments were designed to assess the applicability and utility of the proposed representation for distance information. Performance values on the CityScapes validation set for the “people” class are reported in Table 10.2. Rows 1 to 4 correspond to models trained singularly on each dataset. The least contribution is given by SYNTHIA, which ranked last among all evaluated setups. This dataset was chosen for the presence of human subjects, despite the scarce photorealism of its rendered images. Unexpectedly, the best single-dataset solution appears to be training on the KITTI dataset, which performed much better than using the CityScapes training set itself. This is a first confirmation of the utility of the proposed representation, without which it would have

not be possible to overperform the results obtained with the original training data. In order to verify whether better performance could be obtained simply by using a larger amount of CityScapes images, row 5 describes a model trained on both the CityScapes training set and official test set (which is distinct from the validation set used here for evaluation). The results of this experiment, however, disprove this possibility, ranking at the 9th position. An alternative explanation is that CityScapes offers plenty of images, but with low diversity, and thus it is not useful in building a well-generalizing model.

The second batch of experiments, from row 6 to row 8, involves a joint training on both the CityScapes training set and one of the other datasets. The best result was given by training jointly on the CityScapes and KITTI datasets, which were brought to a compatible format exploiting the proposed representation. Other methods, such as the one proposed by Uhrig et al. [190], were able to obtain good results while training on the CityScapes training set only. One possible explanation can be found in the multi-task objective proposed in their work, which guides the learning process in a different way. In this sense, the proposed approach is an alternative solution when multi-task is not desired or not available.

Given the promising but sub-optimal results of RGB-D People (rows 4 and 8), row 9 shows the results of training jointly on the three best performing sets: CityScapes, KITTI, and RGB-D People. This combination, though, did not improve the model performance, as it ranked at the 3rd place, right after the joint training on KITTI and RGB-D People. Finally, row 10 describes the effect of training on all available datasets. The poor performance is again probably due to the influence of dataset SYNTHIA, as suggested by the results in row 2.

By further training the best performing model (row 7) from 10000 to 45000 iterations, a final MAE equal to 6.24m was reached, on a dataset with average distance 40.39m. Figure 10.7 shows the prediction on an example image obtained with such model. The largest source of error is due to the farthest subject, predicted at 50m instead of 72m, followed by some inaccuracies in the ground truth around people edges. Other visual examples are presented in Figure 10.8.

#### 10.4.4 Applicability of the proposed representation to other methods

The goal of this section is to quantitatively evaluate the benefits of introducing the proposed device-independent representation into a different model for distance estimation. Neven et al. [147] presented a unified framework that predicts pixel-level disparity (i.e.

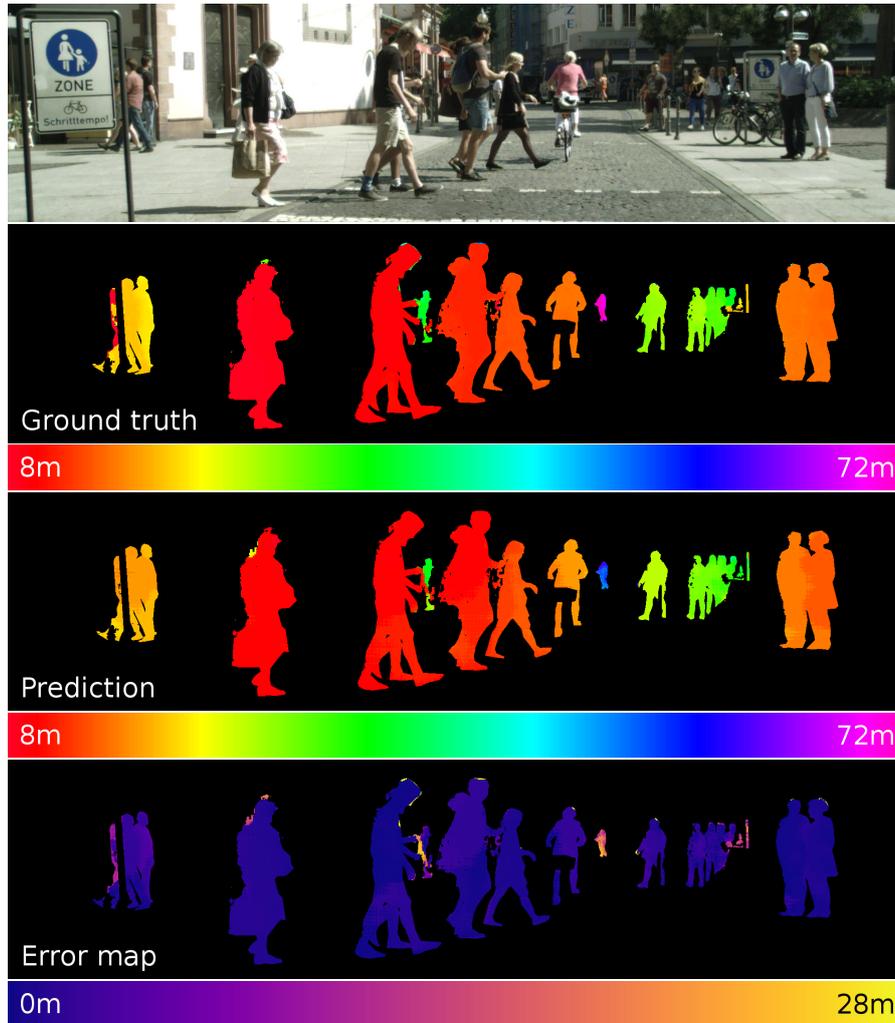


Fig. 10.7 Example per-pixel estimation on a test image with average distance 14.83m. The Mean Absolute Error (MAE) between prediction and ground truth is 1.63m. The visualization and all adopted metrics are limited to people areas, as these first experiments were focused on this particular category of known subjects. Best viewed in color

the horizontal shift necessary to find the same pixel across two images from a stereo rig: a measure correlated to distance information), in a multitask environment that also includes semantic and instance segmentation. In order to do so, the authors trained and evaluated their proposed model on the CityScapes dataset, exploiting its rich set of annotations.

The current experimental setup was first reproduced in the work of [147] by introducing relevance masks in the loss computation, as described in Section 10.2. This allowed the constraining of the training and validation process to only pixels belonging to people areas. This baseline experiment, trained on disparity values, resulted in a  $\text{RMSE}_{\log}$  equal to 0.923, between reference and predicted depth. As a first modification, their disparity representation was replaced at training time with the proposed *real\_size*, resulting in a  $\text{RMSE}_{\log}$  equal to 0.804. The lower error obtained is a first demonstration of the advantage of switching to a size-based representation.

The second intent is to reproduce the experiment on training set expansion from Section 10.4.3 into the method from [147]. The proposed representation, in fact, makes it possible to collect a wider training set by converting different sources of data to a common representation. In order to verify whether this effectively produces a better model, the KITTI training subset was once again introduced as additional training data, as it was shown in Table 10.2 to improve the final estimation performance of the previously tested model. Since this set is not annotated with instance and semantic labels, which are required by the multitask nature of [147], at training time the gradient backpropagation was suppressed from the semantic and instance predictions produced on the KITTI training examples, while keeping those produced from the CityScapes training examples. The model trained in this new environment was finally able to reach an even better  $\text{RMSE}_{\log}$ , equal to 0.715. This result produces yet another indication of the advantage in exploiting the proposed representation for distance information.

### 10.4.5 Multi-class evaluation

In order to further assess the robustness and generalization capability of the proposed representation, the FCN-based neural network was trained and evaluated on other individual semantic classes, namely cars, bikes, and motorbikes. This was achieved by exploiting the relevance masks to exclude regions of the image that do not belong to the chosen subjects. Once again, evaluation was based on the  $\text{RMSE}_{\log}$  obtained on the CityScapes validation set, by either training on the CityScapes only, or by jointly exploiting the KITTI training subset.

## 10.4 Experiments

| Class      | Cardinality     |         |               | RMSE <sub>log</sub> on CSCval |         | % error reduction |
|------------|-----------------|---------|---------------|-------------------------------|---------|-------------------|
|            | Training images |         | Valid. images | Training images               |         |                   |
|            | CSC             | CSC+KIT | CSCval        | CSC                           | CSC+KIT |                   |
| People     | 2498            | 4792    | 441           | 0.677                         | 0.255   | 62.33%            |
| Cars       | 2832            | 4928    | 479           | 0.833                         | 0.366   | 56.02%            |
| Bikes      | 1639            | 3198    | 344           | 0.591                         | 0.323   | 45.35%            |
| Motorbikes | 502             | 1409    | 91            | 0.668                         | 0.577   | 13.68%            |
| Joint      | 2965            | 5273    | 493           | 0.781                         | 0.318   | 59.25%            |

Table 10.3 RMSE<sub>log</sub> errors obtained on different object classes, and evaluated on the validation subset of CityScapes. The last column highlights the percentage error reduction obtained by training on a wider set of images, which is made possible by the proposed representation

Results are reported in Table 10.3, along with the percentage error reduction given by the training set expansion. Some classes appear to benefit more than others: the relative error reduction is in fact strongly correlated with the number of available images, further supporting the thesis on the importance of training cardinality. As a final experiment, the last line of Table 10.3 also reports the results obtained by training and evaluating on all the analyzed semantic classes, i.e. the relevance mask was generated to jointly include people, cars, bikes, and motorbikes. The error reduction given by dataset expansion on this experiment is on par with the trend of all individual classes. This suggests that the model is able to manage multiple classes at the same time, effectively inferring the correct *real\_size* value associated with each different semantic class. Some relevant examples are provided in Figure 10.9

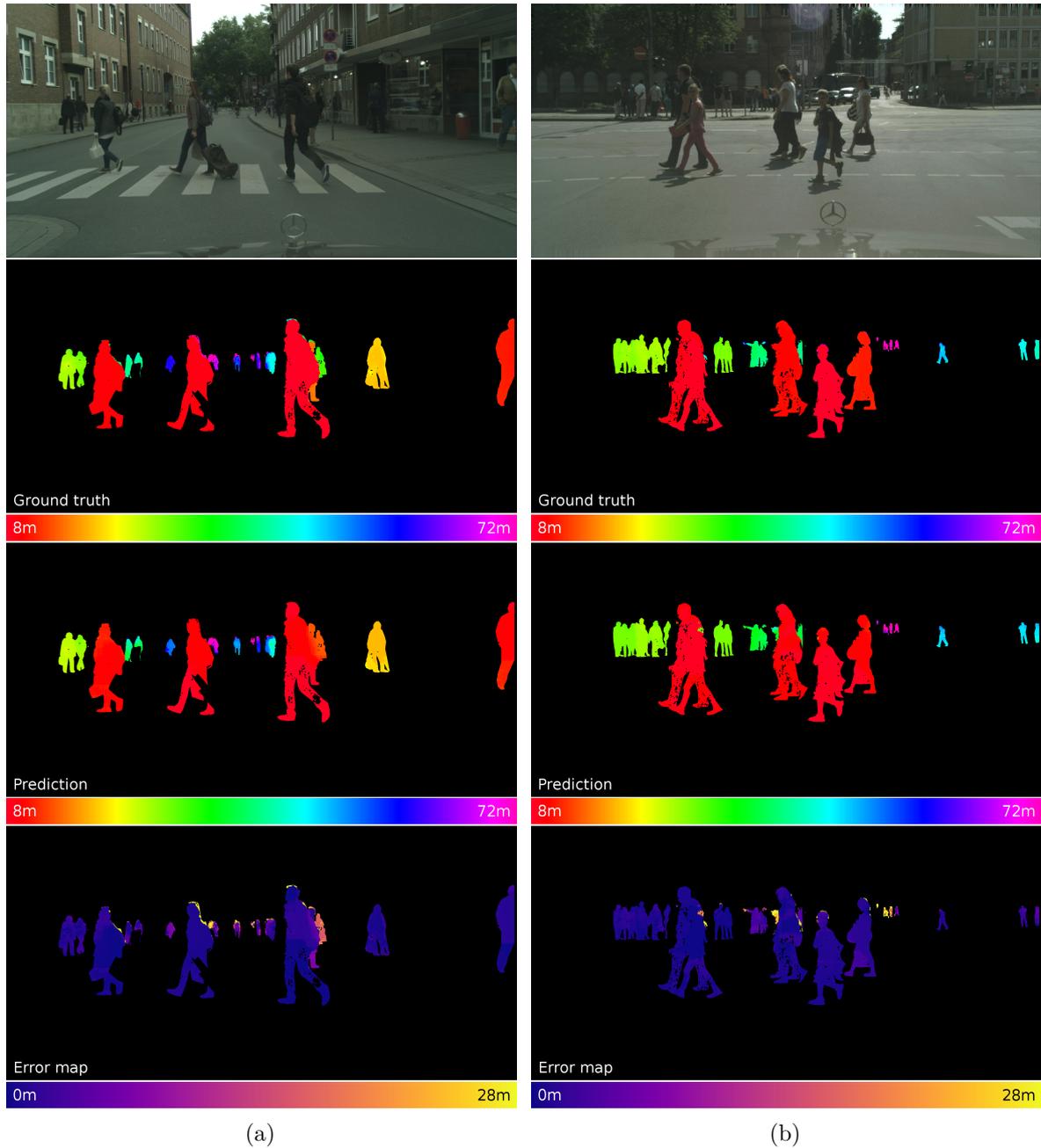


Fig. 10.8 Example predictions on the people class. Values are clipped to the reported range to preserve visual consistency among figures (best viewed in color). a) Average distance annotation: 16.21m. MAE: 2.17m, ARD: 0.10, SRD: 0.91,  $RMSE_{lin}$ : 7.63,  $RMSE_{log}$ : 0.24. b) Average distance annotation: 16.53m. MAE: 1.79m, ARD: 0.10, SRD: 1.07,  $RMSE_{lin}$ : 8.75,  $RMSE_{log}$ : 0.20

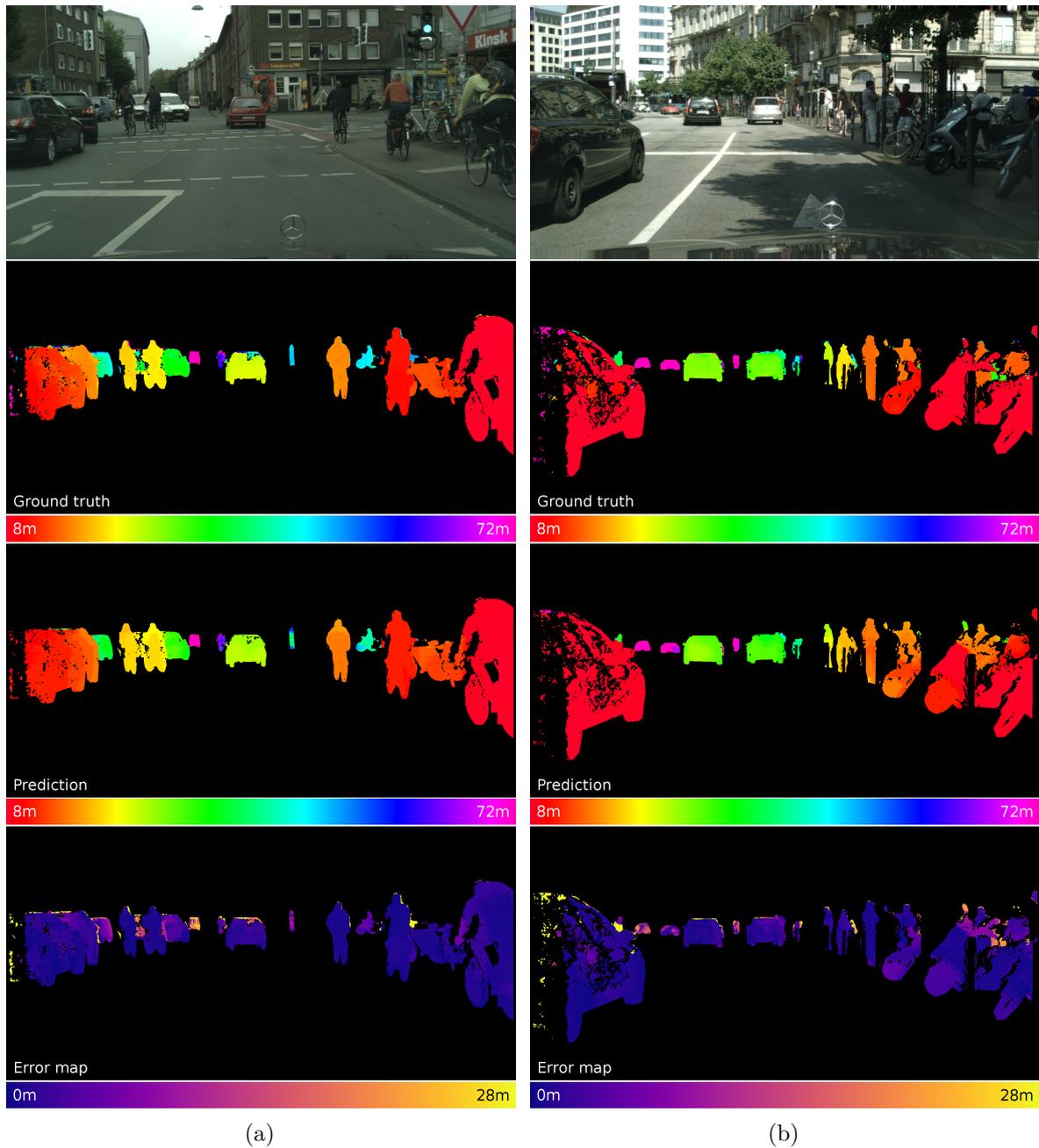


Fig. 10.9 Example predictions on all analyzed semantic classes. Values are clipped to the reported range to preserve visual consistency among figures (best viewed in color). a) Average distance annotation: 24.28m. MAE: 3.84m, ARD: 0.09, SRD: 1.89, RMSE<sub>lin</sub>: 18.00, RMSE<sub>log</sub>: 0.27. b) Average distance annotation: 17.85m. MAE: 3.82m, ARD: 0.13, SRD: 2.31, RMSE<sub>lin</sub>: 25.55, RMSE<sub>log</sub>: 0.31

# Chapter 11

## Conclusions

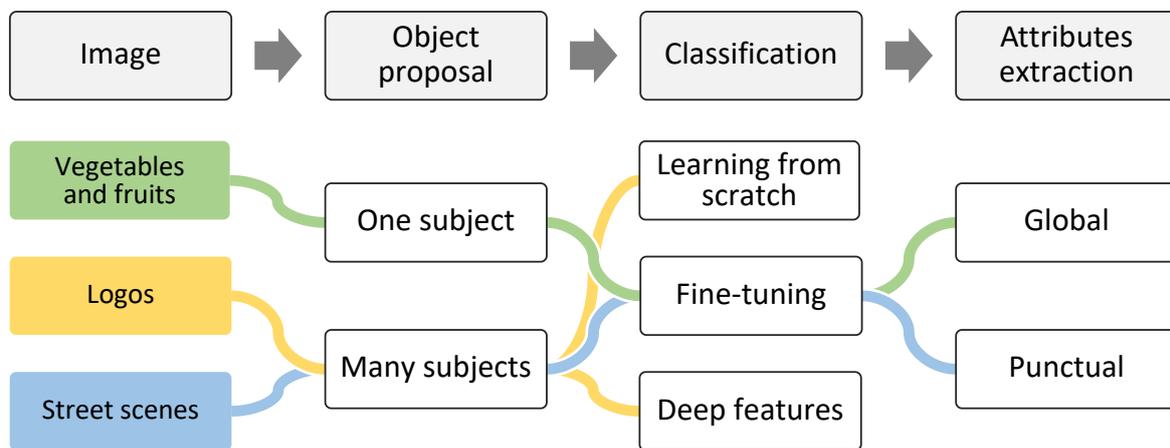


Fig. 11.1 Highlighted uses cases in the proposed pipeline for image description and annotation. The applications involve: vegetables and fruits, logos, and streets.

This thesis covered the problem of automatic description and annotation of complex scenes. The task has been interpreted in terms of a three-step general pipeline, composed of: object proposal, classification, and attributes extraction. For each step, several techniques have been studied, developed, and eventually applied to specific domains with the objective of comparing the proposed solutions against the state of the art. As each domain presents its own set of challenges, three different directions for application have been investigated, aiming at a complete coverage of the nuances of the proposed pipeline.

Vegetables and fruit classification can prove to be a useful resource in the development of internet-of-things-ready appliances. A general purpose salient object estimation technique has been first developed, with the goal of automatically determining the best combination strategy of existing solutions for saliency estimation. The proposed method,

---

based on the joint optimization of genetic programming and backpropagation, allowed to improve performance over all the starting solutions. Salient object detection has then been applied as a preprocessing step for the actual classification of vegetables and fruits, by implementing an automatic cropping mechanism. The recognition itself has been performed by fine-tuning existing networks and exploiting the hierarchical nature of annotated classes. The classification of vegetables has been finally exploited as a proxy to learn the prediction of the scene illuminant, by means of an original training strategy that does not require explicit illuminant ground truth. Experimental results on standard datasets show that the proposed solution can generalize to test images that do not contain the classes seen during the training phase.

Logo recognition is a category of interest due to its prominent presence in the everyday lives of people. At the object proposal stage, an algorithm for hierarchical proposal has been adopted, in view of its proven good compromise between speed and recall. Its configuration was then tuned in order to exploit the specific features of the addressed domain: namely, processing the input image in the HSV color space proved to be the most effective solution, as it allows the subsequent algorithms to take advantage of the bright colors that typically characterize logos. After a preliminary investigation, classification of logos was then addressed by paying attention to two aspects: data and parameters. Regarding data, both real and synthetic data augmentation have been exploited, by collecting a vast dataset and further extending it with realistic transformations. A careful analysis led to a quantification of the impact of both elements, eventually confirming the relevance of the collected dataset. Regarding parameters, an efficient neural model has been developed and enriched with the adoption of smart training strategies. The combination of these factors allowed to reach results that are highly competitive with the state of the art.

Urban street scenes were taken into consideration on grounds of an ever-growing interest in smart cars. Semantic segmentation of streets consists in assigning a categorical label to each pixel in a given input image, with the prediction of classes such as pedestrians, vehicles, and traffic signs. For the envisioned smart-car scenario, high accuracy and fast response time are both fundamental requirements. An original model has been therefore proposed and evaluated, with the specific intention of balancing these two criteria. Secondly, the distance between depicted subjects and the camera has been considered a relevant attribute for a smart-car scenario. Monocular distance estimation has been addressed by developing a device-independent representation of pixel-wise distance information. Such representation allows for richer and heterogeneous datasets

---

to be built by gathering training information from different devices and environments, as proven by careful experimentation.

As a result of this application-driven approach to the problem of image description and annotation, the introduced generic pipeline has been thoroughly investigated under different angles. A direct extension of this would be automatically identifying the nature of the input image in order to select the proper path of analysis in the presented pipeline (e.g. inferring the presence of a single object through saliency estimation, and consequently performing classification without prior knowledge on the possible classes). Additional concepts of description can be taken into consideration, such as image captioning for textual annotation. Finally, as mentioned, the role of data and its representation has been a common background theme throughout the whole work. Illuminant estimation, in particular, has been the strongest reported example of reducing the dependency on expensive ground truth annotation. A further direction for development might involve more direct forms of unsupervised or semi-supervised learning.

# References

- [1] Achanta, R., Hemami, S., Estrada, F., and Susstrunk, S. (2009). Frequency-tuned salient region detection. In *Computer vision and pattern recognition, 2009. cvpr 2009. ieee conference on*, pages 1597–1604. IEEE.
- [2] Adelson, E. H., Anderson, C. H., Bergen, J. R., Burt, P. J., and Ogden, J. M. (1984). Pyramid methods in image processing. *RCA engineer*, 29(6):33–41.
- [3] Akbarinia, A. and Parraga, C. A. (2017). Colour constancy beyond the classical receptive field. *IEEE transactions on pattern analysis and machine intelligence*.
- [4] Albiol, A., Ch, M., Albiol, F., and Torres, L. (2004). Detection of tv commercials. In *2004 IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages iii–541. IEEE.
- [5] Alpert, S., Galun, M., Brandt, A., and Basri, R. (2012). Image segmentation by probabilistic bottom-up aggregation and cue integration. *IEEE transactions on pattern analysis and machine intelligence*, 34(2):315–327.
- [6] Ardizzone, E., Bruno, A., and Mazzola, G. (2011). Visual saliency by keypoints distribution analysis. In *International Conference on Image Analysis and Processing*, pages 691–699. Springer.
- [7] Ardizzone, E., Bruno, A., and Mazzola, G. (2013). Saliency based image cropping. In *International Conference on Image Analysis and Processing*, pages 773–782. Springer.
- [8] Avidan, S. and Shamir, A. (2007). Seam carving for content-aware image resizing. *ACM Trans. Graph.*, 26(3).
- [9] Aytekin, C., Kiranyaz, S., and Gabbouj, M. (2014). Automatic object segmentation by quantum cuts. In *Pattern Recognition (ICPR), 2014 22nd International Conference on*, pages 112–117. IEEE.
- [10] Aytekin, , Ozan, E. C., Kiranyaz, S., and Gabbouj, M. (2015). Visual saliency by extended quantum cuts. In *2015 IEEE International Conference on Image Processing (ICIP)*, pages 1692–1696.
- [11] Badrinarayanan, V., Kendall, A., and Cipolla, R. (2017). Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12):2481–2495.

- 
- [12] 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.
- [13] Battiato, S., Farinella, G. M., Gallo, G., and Giudice, O. (2018). On-board monitoring system for road traffic safety analysis. *Computers in Industry*, 98:208–217.
- [14] Bianco, S., Buzzelli, M., Ciocca, G., and Schettini, R. (2018a). Combining genetic programming and back-propagation for automatic design of saliency estimation models. *Submitted to Information Fusion*.
- [15] Bianco, S., Buzzelli, M., Mazzini, D., and Schettini, R. (2015a). Logo recognition using cnn features. In *Image Analysis and Processing—ICIAP 2015*, pages 438–448. Springer.
- [16] Bianco, S., Buzzelli, M., Mazzini, D., and Schettini, R. (2017a). Deep learning for logo recognition. *Neurocomputing*, 245:23–30.
- [17] Bianco, S., Buzzelli, M., and Schettini, R. (2017b). A fully convolutional network for salient object detection. In *International Conference on Image Analysis and Processing*, pages 82–92. Springer.
- [18] Bianco, S., Buzzelli, M., and Schettini, R. (2018b). Multiscale fully convolutional network for image saliency. *Journal of Electronic Imaging*, 27:27 – 27 – 10.
- [19] Bianco, S., Buzzelli, M., and Schettini, R. (2018c). A unifying representation for pixel-precise distance estimation. *Multimedia Tools and Applications*, pages 1–20.
- [20] Bianco, S., Cadene, R., Celona, L., and Napoletano, P. (2018d). Benchmark analysis of representative deep neural network architectures. *arXiv preprint arXiv:1810.00736*.
- [21] Bianco, S., Celona, L., Napoletano, P., and Schettini, R. (2016). Predicting image aesthetics with deep learning. In *International Conference on Advanced Concepts for Intelligent Vision Systems*, pages 117–125. Springer.
- [22] Bianco, S., Cusano, C., Piccoli, F., and Schettini, R. (2017c). Artistic photo filter removal using convolutional neural networks. *Journal of Electronic Imaging*, 27(1):011004.
- [23] Bianco, S., Cusano, C., and Schettini, R. (2015b). Color constancy using cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 81–89.
- [24] Bianco, S., Mazzini, D., and Schettini, R. (2017d). Deep multibranch neural network for painting categorization. In *International Conference on Image Analysis and Processing*, pages 414–423. Springer.
- [25] Bianco, S., Mazzini, D., and Schettini, R. (2017e). Deep multibranch neural network for painting categorization. In *International Conference on Image Analysis and Processing*, pages 414–423. Springer.

- 
- [26] Bianco, S. and Schettini, R. (2012). Color constancy using faces. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 65–72. IEEE.
- [27] Boia, R., Florea, C., and Florea, L. (2015). Elliptical asift agglomeration in class prototype for logo detection. In *BMVC*, pages 115–1.
- [28] 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.
- [29] Bombonato, L., Camara-Chavez, G., and Silva, P. (2017). Real-time single-shot brand logo recognition. In *Graphics, Patterns and Images (SIBGRAPI), 2017 30th SIBGRAPI Conference on*, pages 134–140. IEEE.
- [30] Borji, A. (2015). What is a salient object? a dataset and a baseline model for salient object detection. *IEEE Transactions on Image Processing*, 24(2):742–756.
- [31] Borji, A., Cheng, M.-M., Jiang, H., and Li, J. (2015). Salient object detection: A benchmark. *IEEE Transactions on Image Processing*, 24(12):5706–5722.
- [32] Bossard, L., Guillaumin, M., and Van Gool, L. (2014). Food-101—mining discriminative components with random forests. In *European Conference on Computer Vision*, pages 446–461. Springer.
- [33] Bramão, I., Faísca, L., Petersson, K. M., and Reis, A. (2012). *The contribution of color to object recognition*. InTech.
- [34] Buzzelli, M., Belotti, F., and Schettini, R. (2018a). Recognition of edible vegetables and fruits for smart home appliances. In *Consumer Electronics Berlin (ICCE-Berlin), 2018. ICCEBerlin 2018. IEEE International Conference on*. IEEE.
- [35] Buzzelli, M., van de Weijer, J., and Schettini, R. (2018b). Learning illuminant estimation from object recognition. In *Image Processing (ICIP), 2018 25th IEEE International Conference on*. IEEE.
- [36] Canziani, A., Paszke, A., and Culurciello, E. (2016). An analysis of deep neural network models for practical applications. *arXiv preprint arXiv:1605.07678*.
- [37] Chakrabarti, A., Hirakawa, K., and Zickler, T. (2012). Color constancy with spatio-spectral statistics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(8):1509–1519.
- [38] Chen, J.-C., Patel, V. M., and Chellappa, R. (2016). Unconstrained face verification using deep cnn features. In *Applications of Computer Vision (WACV), 2016 IEEE Winter Conference on*, pages 1–9. IEEE.
- [39] Chen, L., Papandreou, G., Kokkinos, I., Murphy, K., and Yuille, A. L. (2014). Semantic image segmentation with deep convolutional nets and fully connected crfs. *CoRR*, abs/1412.7062.

- 
- [40] Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., and Yuille, A. L. (2018a). Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848.
- [41] Chen, L.-C., Papandreou, G., Schroff, F., and Adam, H. (2017). Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*.
- [42] Chen, L.-C., Zhu, Y., Papandreou, G., Schroff, F., and Adam, H. (2018b). Encoder-decoder with atrous separable convolution for semantic image segmentation. *arXiv preprint arXiv:1802.02611*.
- [43] Cheng, D., Prasad, D. K., and Brown, M. S. (2014a). Illuminant estimation for color constancy: why spatial-domain methods work and the role of the color distribution. *JOSA A*, 31(5):1049–1058.
- [44] Cheng, M., Mitra, N. J., Huang, X., Torr, P. H. S., and Hu, S. (2015a). Global contrast based salient region detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(3):569–582.
- [45] Cheng, M.-M., Mitra, N. J., Huang, X., and Hu, S.-M. (2014b). Salientshape: Group saliency in image collections. *The Visual Computer*, 30(4):443–453.
- [46] Cheng, M.-M., Mitra, N. J., Huang, X., Torr, P. H., and Hu, S.-M. (2015b). Global contrast based salient region detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(3):569–582.
- [47] Chéron, G., Laptev, I., and Schmid, C. (2015). P-cnn: Pose-based cnn features for action recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 3218–3226.
- [48] Cholakkal, H., Rajan, D., and Johnson, J. (2015). Top-down saliency with locality-constrained contextual sparse coding. In *BMVC*, volume 1, page 5.
- [49] Ciocca, G., Napoletano, P., and Schettini, R. (2015). Food recognition and leftover estimation for daily diet monitoring. In *International Conference on Image Analysis and Processing*, pages 334–341. Springer.
- [50] Ciocca, G., Napoletano, P., and Schettini, R. (2017). Food recognition: a new dataset, experiments, and results. *IEEE journal of biomedical and health informatics*, 21(3):588–598.
- [51] Ciocca, G., Napoletano, P., and Schettini, R. (2018). Cnn-based features for retrieval and classification of food images. *Computer Vision and Image Understanding*, -:-.
- [52] Ciocca, G. and Schettini, R. (2010). Multiple image thumbnailing. In *Digital Photography VI*, volume 7537, page 75370S. International Society for Optics and Photonics.
- [53] Ciurea, F. and Funt, B. (2003). A large image database for color constancy research. In *Color and Imaging Conference*, volume 2003, pages 160–164. Society for Imaging Science and Technology.

- 
- [54] Corchs, S., Ciocca, G., and Schettini, R. (2004). Video summarization using a neurodynamical model of visual attention. In *Multimedia Signal Processing, 2004 IEEE 6th Workshop on*, pages 71–74. IEEE.
- [55] Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., and Schiele, B. (2016a). The cityscapes dataset for semantic urban scene understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- [56] Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., and Schiele, B. (2016b). The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3213–3223.
- [57] Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20(3):273–297.
- [58] Cusano, C., Napoletano, P., and Schettini, R. (2016). Evaluating color texture descriptors under large variations of controlled lighting conditions. *JOSA A*, 33(1):17–30.
- [59] DeCarlo, D. and Santella, A. (2002). Stylization and abstraction of photographs. *ACM Trans. Graph.*, 21(3):769–776.
- [60] Dong, X., Zhang, F., and Shi, P. (2014). A novel approach for face to camera distance estimation by monocular vision. *International Journal of Innovative Computing, Information and Control*, 10(2):659–669.
- [61] Dutton, D. (2009). *The art instinct: Beauty, pleasure, & human evolution*. Oxford University Press, USA.
- [62] 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.
- [63] Eigen, D., Puhrsch, C., and Fergus, R. (2014). Depth map prediction from a single image using a multi-scale deep network. In *Advances in neural information processing systems*, pages 2366–2374.
- [64] Elgammal, A., Duraiswami, R., Harwood, D., and Davis, L. S. (2002). Background and foreground modeling using nonparametric kernel density estimation for visual surveillance. *Proceedings of the IEEE*, 90(7):1151–1163.
- [65] Ens, J. and Lawrence, P. (1993). An investigation of methods for determining depth from focus. *IEEE Transactions on pattern analysis and machine intelligence*, 15(2):97–108.
- [66] Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. (2011). The PASCAL Visual Object Classes Challenge 2011 (VOC2011) Results. <http://www.pascal-network.org/challenges/VOC/voc2011/workshop/index.html>.

- 
- [67] Farinella, G. M., Moltisanti, M., and Battiato, S. (2014). Classifying food images represented as bag of textons. In *Image Processing (ICIP), 2014 IEEE International Conference on*, pages 5212–5216. IEEE.
- [68] Felzenszwalb, P. F., Girshick, R. B., McAllester, D., and Ramanan, D. (2010). Object detection with discriminatively trained part-based models. *IEEE transactions on pattern analysis and machine intelligence*, 32(9):1627–1645.
- [69] Felzenszwalb, P. F. and Huttenlocher, D. P. (2004). Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181.
- [70] Fitzsimons, G. M., Chartrand, T. L., and Fitzsimons, G. J. (2008). Automatic effects of brand exposure on motivated behavior: how apple makes you “think different”. *Journal of consumer research*, 35(1):21–35.
- [71] Forsyth, D. A. (1990). A novel algorithm for color constancy. *International Journal of Computer Vision*, 5(1):5–35.
- [72] Funt, B. and Mosny, M. (2012). Removing outliers in illumination estimation. In *Color and Imaging Conference*, volume 2012, pages 105–110. Society for Imaging Science and Technology.
- [73] Furda, A. and Vlacic, L. (2011). Enabling safe autonomous driving in real-world city traffic using multiple criteria decision making. *IEEE Intelligent Transportation Systems Magazine*, 3(1):4–17.
- [74] Gao, D., Han, S., and Vasconcelos, N. (2009). Discriminant saliency, the detection of suspicious coincidences, and applications to visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(6):989–1005.
- [75] Gao, S.-B., Yang, K.-F., Li, C.-Y., and Li, Y.-J. (2015a). Color constancy using double-opponency. *IEEE transactions on pattern analysis and machine intelligence*, 37(10):1973–1985.
- [76] Gao, Y., Beijbom, O., Zhang, N., and Darrell, T. (2016). Compact bilinear pooling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 317–326.
- [77] Gao, Y., Shi, M., Tao, D., and Xu, C. (2015b). Database saliency for fast image retrieval. *IEEE Transactions on Multimedia*, 17(3):359–369.
- [78] 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.
- [79] Gehler, P. V., Rother, C., Blake, A., Minka, T., and Sharp, T. (2008). Bayesian color constancy revisited. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE.
- [80] Geiger, A., Lenz, P., Stiller, C., and Urtasun, R. (2013). Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237.

- 
- [81] Gijssenij, A., Gevers, T., and Van De Weijer, J. (2011). Computational color constancy: Survey and experiments. *IEEE Transactions on Image Processing*, 20(9):2475–2489.
- [82] Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 580–587. IEEE.
- [83] Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256.
- [84] Godard, C., Mac Aodha, O., and Brostow, G. J. (2016). Unsupervised monocular depth estimation with left-right consistency. *arXiv preprint arXiv:1609.03677*.
- [85] Gossan, S. and Ott, C. (2012). Methods of measuring astronomical distances.
- [86] Goyal, Y., Khot, T., Summers-Stay, D., Batra, D., and Parikh, D. (2017). Making the v in vqa matter: Elevating the role of image understanding in visual question answering. In *CVPR*, volume 1, page 3.
- [87] Grace, K., Salvatier, J., Dafoe, A., Zhang, B., and Evans, O. (2018). When will ai exceed human performance? evidence from ai experts. *Journal of Artificial Intelligence Research*, 62:729–754.
- [88] Guo, C. and Zhang, L. (2010). A novel multiresolution spatiotemporal saliency detection model and its applications in image and video compression. *IEEE Transactions on Image Processing*, 19(1):185–198.
- [89] Hagbi, N., Bergig, O., El-Sana, J., and Billinghurst, M. (2011). Shape recognition and pose estimation for mobile augmented reality. *Visualization and Computer Graphics, IEEE Transactions on*, 17(10):1369–1379.
- [90] Harkness, L. (1977). Chameleons use accommodation cues to judge distance. *Nature*, 267(5609):346–349.
- [91] He, K., Zhang, X., Ren, S., and Sun, J. (2016a). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.
- [92] He, K., Zhang, X., Ren, S., and Sun, J. (2016b). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.
- [93] Hirschmuller, H. (2005). Accurate and efficient stereo processing by semi-global matching and mutual information. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 807–814. IEEE.
- [94] Hochberg, C. B. and Hochberg, J. E. (1952). Familiar size and the perception of depth. *The Journal of Psychology*, 34(1):107–114.

- 
- [95] Hoiem, D., Efros, A. A., and Hebert, M. (2008). Putting objects in perspective. *International Journal of Computer Vision*, 80(1):3–15.
- [96] Hong, D., Tavanapong, W., Wong, J., Oh, J., and De Groen, P. C. (2014). 3d reconstruction of virtual colon structures from colonoscopy images. *Computerized Medical Imaging and Graphics*, 38(1):22–33.
- [97] 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.
- [98] Hou, Q., Cheng, M., Hu, X., Borji, A., Tu, Z., and Torr, P. H. S. (2018). Deeply supervised salient object detection with short connections. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, page In press.
- [99] Hou, Q., Cheng, M.-M., Hu, X., Borji, A., Tu, Z., and Torr, P. (2017a). Deeply supervised salient object detection with short connections. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 5300–5309. IEEE.
- [100] Hou, S., Feng, Y., and Wang, Z. (2017b). Vegfru: A domain-specific dataset for fine-grained visual categorization. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 541–549. IEEE.
- [101] Howard, I. P. and Rogers, B. J. (1995). *Binocular vision and stereopsis*. Oxford University Press, USA.
- [102] 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*.
- [103] Itti, L., Koch, C., and Niebur, E. (1998). A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on pattern analysis and machine intelligence*, 20(11):1254–1259.
- [104] Jelodar, A. B., Salekin, M. S., and Sun, Y. (2018). Identifying object states in cooking-related images. *arXiv preprint arXiv:1805.06956*.
- [105] Jiang, B., Zhang, L., Lu, H., Yang, C., and Yang, M.-H. (2013a). Saliency detection via absorbing markov chain. In *Proceedings of the IEEE international conference on computer vision*, pages 1665–1672.
- [106] Jiang, H., Wang, J., Yuan, Z., Wu, Y., Zheng, N., and Li, S. (2013b). Salient object detection: A discriminative regional feature integration approach. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2083–2090.
- [107] 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.
- [108] Joze, H. R. V. and Drew, M. S. (2014). Exemplar-based color constancy and multiple illumination. *IEEE transactions on pattern analysis and machine intelligence*, 36(5):860–873.

- 
- [109] Kim, J., Han, D., Tai, Y., and Kim, J. (2016). Salient region detection via high-dimensional color transform and local spatial support. *IEEE Transactions on Image Processing*, 25(1):9–23.
- [110] 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.
- [111] Koza, J. R. (1994). Genetic programming as a means for programming computers by natural selection. *Statistics and computing*, 4(2):87–112.
- [112] Krizhevsky, A. and Hinton, G. (2010). Convolutional deep belief networks on cifar-10. *Unpublished manuscript*, 40:7.
- [113] 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.
- [114] Ladicky, L., Shi, J., and Pollefeys, M. (2014). Pulling things out of perspective. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 89–96.
- [115] LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436.
- [116] Lee, G., Tai, Y.-W., and Kim, J. (2016). Deep saliency with encoded low level distance map and high level features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 660–668.
- [117] Lempitsky, V. and Zisserman, A. (2010). Learning to count objects in images. In *Advances in neural information processing systems*, pages 1324–1332.
- [118] Li, A., She, X., and Sun, Q. (2013a). Color image quality assessment combining saliency and fsim. In *Fifth International Conference on Digital Image Processing (ICDIP 2013)*, volume 8878, page 88780I. International Society for Optics and Photonics.
- [119] Li, B., Shen, C., Dai, Y., van den Hengel, A., and He, M. (2015). Depth and surface normal estimation from monocular images using regression on deep features and hierarchical crfs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1119–1127.
- [120] Li, G. and Yu, Y. (2015). Visual saliency based on multiscale deep features. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5455–5463.
- [121] Li, G. and Yu, Y. (2016). Deep contrast learning for salient object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 478–487.
- [122] Li, Q., Zhou, Y., and Yang, J. (2011). Saliency based image segmentation. In *2011 International Conference on Multimedia Technology*, pages 5068–5071.

- 
- [123] Li, X., Lu, H., Zhang, L., Ruan, X., and Yang, M. (2013b). Saliency detection via dense and sparse reconstruction. In *2013 IEEE International Conference on Computer Vision*, pages 2976–2983.
- [124] Li, X., Zhao, L., Wei, L., Yang, M., Wu, F., Zhuang, Y., Ling, H., and Wang, J. (2016). Deepsaliency: Multi-task deep neural network model for salient object detection. *IEEE Transactions on Image Processing*, 25(8):3919–3930.
- [125] Li, Y., Hou, X., Koch, C., Rehg, J. M., and Yuille, A. L. (2014). The secrets of salient object segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 280–287.
- [126] Lin, G., Shen, C., Van Den Hengel, A., and Reid, I. (2016). Efficient piecewise training of deep structured models for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3194–3203.
- [127] Liu, F., Shen, C., Lin, G., and Reid, I. (2016). Learning depth from single monocular images using deep convolutional neural fields. *IEEE transactions on pattern analysis and machine intelligence*, 38(10):2024–2039.
- [128] Liu, N. and Han, J. (2016). Dhsnet: Deep hierarchical saliency network for salient object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 678–686.
- [129] Liu, N. and Han, J. (2018). A deep spatial contextual long-term recurrent convolutional network for saliency detection. *IEEE Transactions on Image Processing*, 27(7):3264–3274.
- [130] Liu, T., Yuan, Z., Sun, J., Wang, J., Zheng, N., Tang, X., and Shum, H.-Y. (2011). Learning to detect a salient object. *IEEE Transactions on Pattern analysis and machine intelligence*, 33(2):353–367.
- [131] Liu, Z., Zou, W., and Le Meur, O. (2014a). Saliency tree: A novel saliency detection framework. *IEEE Transactions on Image Processing*, 23(5):1937–1952.
- [132] Liu, Z., Zou, W., and Meur, O. L. (2014b). Saliency tree: A novel saliency detection framework. *IEEE Transactions on Image Processing*, 23(5):1937–1952.
- [133] Long, J., Shelhamer, E., and Darrell, T. (2015a). Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440.
- [134] Long, J., Shelhamer, E., and Darrell, T. (2015b). Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440.
- [135] Lou, Z., Gevers, T., Hu, N., Lucassen, M. P., et al. (2015). Color constancy by deep learning. In *BMVC*, pages 76–1.
- [136] Mahadevan, V. and Vasconcelos, N. (2009). Saliency-based discriminant tracking. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1007–1013.

- 
- [137] Mai, L., Niu, Y., and Liu, F. (2013). Saliency aggregation: A data-driven approach. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1131–1138.
- [138] Margolin, R., Zelnik-Manor, L., and Tal, A. (2013). Saliency for image manipulation. *The Visual Computer*, 29(5):381–392.
- [139] Marotta, J., Perrot, T., Nicolle, D., Servos, P., and Goodale, M. (1995). Adapting to monocular vision: grasping with one eye. *Experimental Brain Research*, 104(1):107–114.
- [140] Martin, D., Fowlkes, C., Tal, D., and Malik, J. (2001). A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 2, pages 416–423. IEEE.
- [141] Mazzini, D., Buzzelli, M., Pau, D. P., and Schettini, R. (2018). A cnn architecture for efficient semantic segmentation of street scenes. In *Consumer Electronics Berlin (ICCE-Berlin), 2018. ICCEBerlin 2018. IEEE International Conference on*. IEEE.
- [142] Mendelson, A. L. and Papacharissi, Z. (2010). Look at us: Collective narcissism in college student facebook photo galleries. *The networked self: Identity, community and culture on social network sites*, 1974:1–37.
- [143] 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.
- [144] Mitri, S., Frintrop, S., Pervözl, K., Surmann, H., and Nüchter, A. (2005). Robust object detection at regions of interest with an application in ball recognition. In *IEEE international conference on robotics and automation*, volume 1, page 125. IEEE; 1999.
- [145] Mureşan, H. and Oltean, M. (2017). Fruit recognition from images using deep learning. *arXiv preprint arXiv:1712.00580*.
- [146] Navalpakkam, V. and Itti, L. (2006). An integrated model of top-down and bottom-up attention for optimizing detection speed. In *null*, pages 2049–2056. ieee.
- [147] Neven, D., De Brabandere, B., Georgoulis, S., Proesmans, M., and Van Gool, L. (2017). Fast scene understanding for autonomous driving. *arXiv preprint arXiv:1708.02550*.
- [148] Oh, S. W. and Kim, S. J. (2017). Approaching the computational color constancy as a classification problem through deep learning. *Pattern Recognition*, 61:405–416.
- [149] 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.
- [150] Otsu, N. (1979). A threshold selection method from gray-level histograms. *IEEE transactions on systems, man, and cybernetics*, 9(1):62–66.

- 
- [151] Ouerhani, N., Bracamonte, J., Hügli, H., Ansorge, M., and Pellandini, F. (2001). Adaptive color image compression based on visual attention. In *Proceedings of the 11th International Conference on Image Analysis and Processing (ICIAP)*, volume 11, pages 416–421. Institute of Electrical and Electronics Engineers (IEEE).
- [152] Paszke, A., Chaurasia, A., Kim, S., and Culurciello, E. (2016). Enet: A deep neural network architecture for real-time semantic segmentation. *arXiv preprint arXiv:1606.02147*.
- [153] Paulin, M., Revaud, J., Harchaoui, Z., Perronnin, F., and Schmid, C. (2014). Transformation pursuit for image classification. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 3646–3653. IEEE.
- [154] Perazzi, F., Krähenbühl, P., Pritch, Y., and Hornung, A. (2012). Saliency filters: Contrast based filtering for salient region detection. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 733–740. IEEE.
- [155] Pitaloka, D. A., Wulandari, A., Basaruddin, T., and Liliana, D. Y. (2017). Enhancing cnn with preprocessing stage in automatic emotion recognition. *Procedia Computer Science*, 116:523–529.
- [156] Pohlen, T., Hermans, A., Mathias, M., and Leibe, B. (2017). Full-resolution residual networks for semantic segmentation in street scenes. *arXiv preprint*.
- [157] Poslad, S. (2011). *Ubiquitous computing: smart devices, environments and interactions*. John Wiley & Sons.
- [158] Prados, E. and Faugeras, O. (2006). Shape from shading. *Handbook of mathematical models in computer vision*, pages 375–388.
- [159] Psylos, 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.
- [160] Ranftl, R., Vineet, V., Chen, Q., and Koltun, V. (2016). Dense monocular depth estimation in complex dynamic scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4058–4066.
- [161] Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788.
- [162] 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.
- [163] Ren, Z., Gao, S., Chia, L.-T., and Tsang, I. W.-H. (2014). Region-based saliency detection and its application in object recognition. *IEEE Trans. Circuits Syst. Video Techn.*, 24(5):769–779.

- 
- [164] 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.
- [165] Riesenhuber, M. and Poggio, T. (1999). Hierarchical models of object recognition in cortex. *Nature neuroscience*, 2(11):1019.
- [166] Rodrigues, D. G., Grenader, E., Nos, F. d. S., Dall’Agnol, M. d. S., Hansen, T. E., and Weibel, N. (2013). Motiondraw: a tool for enhancing art and performance using kinect. In *CHI’13 Extended Abstracts on Human Factors in Computing Systems*, pages 1197–1202. ACM.
- [167] 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.
- [168] 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.
- [169] Romera, E., Alvarez, J. M., Bergasa, L. M., and Arroyo, R. (2018). Erfnet: Efficient residual factorized convnet for real-time semantic segmentation. *IEEE Transactions on Intelligent Transportation Systems*, 19(1):263–272.
- [170] Ros, G., Sellart, L., Materzynska, J., Vazquez, D., and Lopez, A. M. (2016). The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3234–3243.
- [171] Rother, C., Kolmogorov, V., and Blake, A. (2004). Grabcut: Interactive foreground extraction using iterated graph cuts. In *ACM transactions on graphics (TOG)*, volume 23, pages 309–314. ACM.
- [172] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, 323(6088):533.
- [173] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252.
- [174] Scharstein, D. and Szeliski, R. (2003). High-accuracy stereo depth maps using structured light. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 1, pages I–I. IEEE.
- [175] Schneider, W. and Shiffrin, R. M. (1977). Controlled and automatic human information processing: I. detection, search, and attention. *Psychological review*, 84(1):1.
- [176] Sethuraman, N. and Smith, L. B. (2010). Cross-linguistic differences in talking about scenes. *Journal of pragmatics*, 42(11):2978.

- 
- [177] 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.
- [178] Shi, L. (2000). Re-processed version of the gehler color constancy dataset of 568 images. <http://www.cs.sfu.ca/~color/data/>.
- [179] Shiffrin, R. M. and Schneider, W. (1977). Controlled and automatic human information processing: Ii. perceptual learning, automatic attending and a general theory. *Psychological review*, 84(2):127.
- [180] Sill, J., Takács, G., Mackey, L., and Lin, D. (2009). Feature-weighted linear stacking. *arXiv preprint arXiv:0911.0460*.
- [181] Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.
- [182] Smet, K. A., Zhai, Q., Luo, M. R., and Hanselaer, P. (2017). Study of chromatic adaptation using memory color matches, part i: neutral illuminants. *Optics express*, 25(7):7732–7748.
- [183] Spinello, L. and Arras, K. O. (2011). People detection in rgb-d data. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 3838–3843. IEEE.
- [184] 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.
- [185] Subbarao, M. and Surya, G. (1994). Depth from defocus: a spatial domain approach. *International Journal of Computer Vision*, 13(3):271–294.
- [186] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A., et al. (2015). Going deeper with convolutions. *Cvpr*.
- [187] Teng, C.-Y., Lin, Y.-R., and Adamic, L. A. (2012). Recipe recommendation using ingredient networks. In *Proceedings of the 4th Annual ACM Web Science Conference*, pages 298–307. ACM.
- [188] Torralba, A. and Oliva, A. (2002). Depth estimation from image structure. *IEEE Transactions on pattern analysis and machine intelligence*, 24(9):1226–1238.
- [189] Trembl, M., Arjona-Medina, J., Unterthiner, T., Durgesh, R., Friedmann, F., Schuberth, P., Mayr, A., Heusel, M., Hofmarcher, M., Widrich, M., et al. (2016). Speeding up semantic segmentation for autonomous driving. In *MLITS, NIPS Workshop*.
- [190] Uhrig, J., Cordts, M., Franke, U., and Brox, T. (2016). Pixel-level encoding and depth layering for instance-level semantic labeling. In *German Conference on Pattern Recognition*, pages 14–25. Springer International Publishing.
- [191] 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.

- 
- [192] 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.
- [193] Wang, J., Jiang, H., Yuan, Z., Cheng, M.-M., Hu, X., and Zheng, N. (2017). Salient object detection: A discriminative regional feature integration approach. *International Journal of Computer Vision*, 123(2):251–268.
- [194] Wang, L., Wang, L., Lu, H., Zhang, P., and Ruan, X. (2016). Saliency detection with recurrent fully convolutional networks. In *European Conference on Computer Vision*, pages 825–841. Springer.
- [195] Wang, L., Wang, L., Lu, H., Zhang, P., and Ruan, X. (2018). Salient object detection with recurrent fully convolutional networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, page In press.
- [196] Wang, Z., Bovik, A. C., Sheikh, H. R., and Simoncelli, E. P. (2004). Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612.
- [197] Wedel, A., Franke, U., Klappstein, J., Brox, T., Cremers, D., et al. (2006). Realtime depth estimation and obstacle detection from monocular video. *Lecture notes in computer science*, 4174:475.
- [198] Wei, Y., Wen, F., Zhu, W., and Sun, J. (2012). Geodesic saliency using background priors. In *European conference on computer vision*, pages 29–42. Springer.
- [199] Wong, L. and Low, K. (2011). Saliency retargeting: An approach to enhance image aesthetics. In *2011 IEEE Workshop on Applications of Computer Vision (WACV)*, pages 73–80.
- [200] Wu, Z., Shen, C., and Hengel, A. v. d. (2016). Wider or deeper: Revisiting the resnet model for visual recognition. *arXiv preprint arXiv:1611.10080*.
- [201] Xie, S. and Tu, Z. (2015). Holistically-nested edge detection. In *Proceedings of the IEEE international conference on computer vision*, pages 1395–1403.
- [202] Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., Zemel, R., and Bengio, Y. (2015). Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057.
- [203] Yan, Q., Xu, L., Shi, J., and Jia, J. (2013a). Hierarchical saliency detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1155–1162.
- [204] Yan, Q., Xu, L., Shi, J., and Jia, J. (2013b). Hierarchical saliency detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1155–1162.
- [205] Yang, C., Zhang, L., Lu, H., Ruan, X., and Yang, M. (2013a). Saliency detection via graph-based manifold ranking. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3166–3173.

- 
- [206] Yang, C., Zhang, L., Lu, H., Ruan, X., and Yang, M.-H. (2013b). Saliency detection via graph-based manifold ranking. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3166–3173.
- [207] Yonas, A., Petterson, L., and Granrud, C. E. (1982). Infants’ sensitivity to familiar size as information for distance. *Child Development*, pages 1285–1290.
- [208] Yu, F. and Koltun, V. (2016). Multi-scale context aggregation by dilated convolutions. In *International Conference on Learning Representations (ICLR)*.
- [209] Yu, F., Koltun, V., and Funkhouser, T. (2017). Dilated residual networks. In *Computer Vision and Pattern Recognition (CVPR)*.
- [210] Zhang, J., Sclaroff, S., Lin, Z., Shen, X., Price, B., and Mech, R. (2015a). Minimum barrier salient object detection at 80 fps. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1404–1412.
- [211] Zhang, J., Sclaroff, S., Lin, Z., Shen, X., Price, B., and Mech, R. (2015b). Minimum barrier salient object detection at 80 fps. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1404–1412.
- [212] Zhang, Z. (2012). Microsoft kinect sensor and its effect. *IEEE multimedia*, 19(2):4–10.
- [213] Zhao, H., Qi, X., Shen, X., Shi, J., and Jia, J. (2017a). Icnets for real-time semantic segmentation on high-resolution images. *arXiv preprint arXiv:1704.08545*.
- [214] Zhao, H., Shi, J., Qi, X., Wang, X., and Jia, J. (2017b). Pyramid scene parsing network. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2881–2890.
- [215] Zhao, R., Ouyang, W., Li, H., and Wang, X. (2015a). Saliency detection by multi-context deep learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1265–1274.
- [216] Zhao, R., Ouyang, W., Li, H., and Wang, X. (2015b). Saliency detection by multi-context deep learning. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1265–1274.
- [217] Zheng, S., Jayasumana, S., Romera-Paredes, B., Vineet, V., Su, Z., Du, D., Huang, C., and Torr, P. H. (2015). Conditional random fields as recurrent neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1529–1537.
- [218] Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., and Torralba, A. (2016). Learning deep features for discriminative localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2921–2929.
- [219] Zhu, W., Liang, S., Wei, Y., and Sun, J. (2014a). Saliency optimization from robust background detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2814–2821.

- [220] Zhu, W., Liang, S., Wei, Y., and Sun, J. (2014b). Saliency optimization from robust background detection. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2814–2821.
- [221] Zoph, B., Vasudevan, V., Shlens, J., and Le, Q. V. (2017). Learning transferable architectures for scalable image recognition. *arXiv preprint arXiv:1707.07012*.