UNIVERSITÀ DEGLI STUDI DI CATANIA
Dipartimento di Matematica e Informatica
Dottorato di Ricerca in Matematica e Informatica XXXI Ciclo

Daniele Di Mauro

Scene Understanding for Parking Spaces Management

Tesi di Dottorato di Ricerca

Prof. Giovanni Maria Farinella

Anno Accademico 2017 - 2018

"*Everything which distinguishes man from the animals depends upon this ability to volatilize perceptual metaphors in a schema, and thus to dissolve an image into a concept.*"

F. Nietzsche in "On Truth and Lies in a Nonmoral Sense"

Abstract

The major part of world-wide population moved to urban areas. After such process many issues of major cities have worsened, e.g. air pollution, traffic, security. The increase of security cameras and the improvements of Computer Vision algorithm can be a good solution for many of those problems.
The work in this thesis was started after a grant by Park Smart s.r.l., a company located in Catania, which believes that Computer Vision can be the answer for parking space management. The main problem the company has to face is to find a fast way to deploy working solutions, lowering the labeling effort to the minimum, across different scene, cities, parking areas. During the three years of doctoral studies we have tried to solve the problem through the use of various methods such as Semi-Supervised Learning, Counting and Scene Adaptation through Image Classification, Object Detection and Semantic Segmentation.
Semi-Supervised classification was the first approach used to decrease labeling effort for fast deployment. Methods based on counting objects, like cars and parking spots, were analyzed as second solution. To gain full knowledge of the scene we focused on Semantic Segmentation and the use of Generative Adversarial Networks in order to find a viable way to reach good Scene Adaptation results comparable to state-of-the-art methods.

Acknowledgements

I would like to thank my Ph.D. supervisor Prof. Giovanni Maria Farinella (University of Catania) for his advice, as well as Prof. Sebastiano Battiato (University of Catania), Dr. Antonino Furnari (University of Catania), Giuseppe Patané (Park Smart) and Dr. Marco Moltisanti (Park Smart) for their support and suggestions.

Contents

Abstract ii

Acknowledgements iii

1 Introduction 1
- 1.1 Parking Management and Computer Vision . . . 3
- 1.2 Aims and Findings . . . 5
- 1.3 Contribution . . . 5
- 1.4 Thesis Outline . . . 6

2 Computer Vision: Background and State of the Art 8
- 2.1 Classification . . . 8
  - 2.1.1 Gradient-based learning applied to document recognition (LeNet) 9
  - 2.1.2 ImageNet Classification with Deep Convolutional Neural Networks (AlexNet) . . . 10
  - 2.1.3 Going Deeper with Convolutions (GoogLeNet) . . . 11
  - 2.1.4 Very Deep Convolutional Networks for Large-Scale Image Recognition (VGG) . . . 11
  - 2.1.5 Deep Residual Learning for Image Recognition (Resnet) . . . 12
- 2.2 Object Detection . . . 12
  - 2.2.1 Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks (Faster RCNN) . . . 14
  - 2.2.2 You Only Look Once: Unified, Real-Time Object Detection (YOLO) . . . 15
- 2.3 Semantic Segmentation . . . 16
  - 2.3.1 SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation . . . 16
  - 2.3.2 Fully Convolutional Networks for Semantic Segmentation (FCN) 17

2.3.3 DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs . 18
2.3.4 Pyramid Scene Parsing Network (PSPnet) 18
2.4 Generative Models and Applications 19
2.4.1 Generative Adversarial Networks 19
2.4.2 Image Translation 20
2.4.3 Transfer Learning and Domain Adaptation 21
2.4.4 Domain Adaptation for Semantic Segmentation 22

3 Empty vs Not-empty Parking Space Classification 23
3.1 Park Smart: The Overall System at Glance 23
3.2 Classification of parking stalls 25
3.3 Discussions 27

4 Semisupervised Learning Approach Using Pseudolabels 28
4.1 Methods 29
4.1.1 Deep Convolutional Neural Networks and Fine-Tuning 30
4.1.2 Pseudo-Label: The Simple and Efficient Semi-Supervised Learning Method for Deep Neural Networks 30
4.2 Dataset 31
4.2.1 PKLot Dataset 31
4.2.2 PSD Dataset 31
4.3 Experiments and Results 33
4.4 Discussions 35

5 Counting Cars And Non-empty Parking Space In a Monitored Area 37
5.1 Methods 38
5.1.1 Stall-Based Occupancy Estimation 38
5.1.2 Stall-Free Occupancy Estimation 40
Temporal Smoothing 42
5.2 Dataset 42
5.3 Experimental Settings 46
5.3.1 Optimization of Thresholds and Seletion of RoI 47

5.3.2 Evaluation Measures . . . 49
5.4 Results . . . 50
5.5 Discussions . . . 54

6 Scene Adaptation for Semantic Segmentation using Adversarial Learning 56
6.1 Method . . . 58
6.1.1 Network Architecture . . . 58
Semantic Segmentation Network . . . 59
Generative Network . . . 59
Discriminator Network . . . 59
6.1.2 Loss Functions . . . 60
Semantic Segmentation Loss . . . 60
Reconstruction Loss . . . 60
Adversarial Loss . . . 61
Overall Loss . . . 61
6.2 Datasets . . . 62
6.3 Experimental Settings . . . 63
6.4 Results . . . 64
6.4.1 Comparison with "the state of the art" . . . 64
6.4.2 Improvement of Performance on the Source Domain . . . 70
6.4.3 Geometric Warp Baseline . . . 70
6.4.4 Ablation Study . . . 71
6.5 Discussions . . . 71

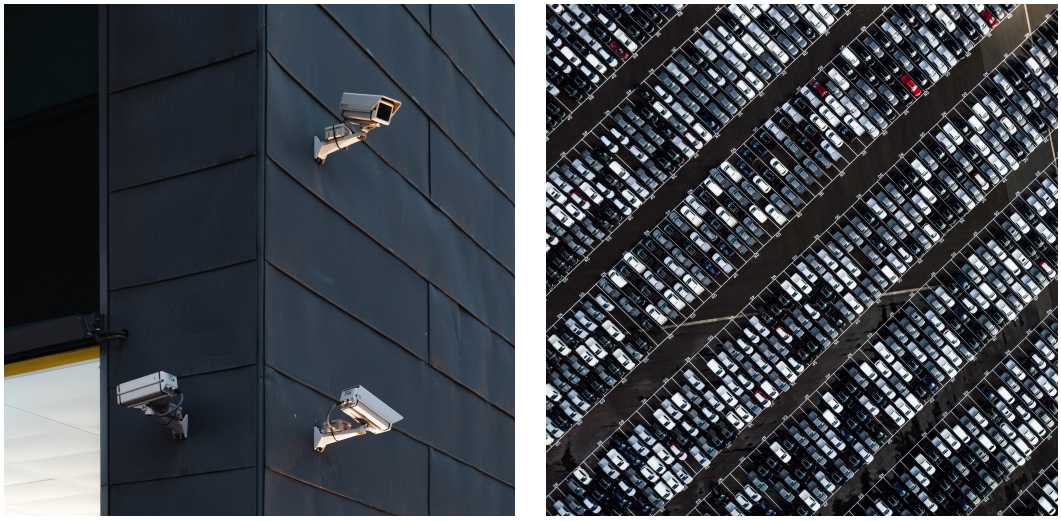
7 Conclusions 73
7.1 Future Directions . . . 74

Bibliography 75

Chapter 1

Introduction

The social context where we live is changing rapidly. Recent studies show that more than half of global population lives in urban areas [1]. Cities are becoming more and more messy and disordered due to such huge number of inhabitants. These megacities are affected by new problems in different fields. To face this kind of problems, leveraging new technologies to ease their solutions, the scientific community and the governments elaborated the concept of *Smart Cities*.
A Smart City is a city where, with a massive use of ICT solutions, classic problems such as traffic, health monitoring, mobility, efficient governance, etc. are tackled in an innovative fashion. Moreover, in the last few years there has been a significant interest about the Internet of Things paradigm [2]. The IoT has been defined as "a global infrastructure for the information society, enabling advanced services by interconnecting (physical and virtual) things based on existing and evolving interoperable information and communication technologies" [3]. Thus Smart Cities are intrinsically correlated to the implementation of an IoT framework. In such a scenario, sensors are distributed in the area of the cities and generally connected via cloud, to exploit the computational power of remote machines. This model though is subject to the limitation brought by the availability of a high speed connection.
Recently, major companies have started to focus on a new paradigm, known as *edge computing* [4]. In this model, the computation is "moved" from inside the cloud to its borders, making use of distributed processing units. Thus the IoT device is transformed from a mere sensor to an intelligent unit. Smart cities is thus one of the new frontier of the Computer Vision community thanks to increase of security cameras (see Figure 1.1 (a)) and the improvements of Computer Vision algorithms. Among the different applications of Computer Vision to Smart Cities we can recall



(a) (b)

Figure 1.1: (a) Surveillance cameras spotted at Barcelona's design museum. (Photo by Ennio Dybeli on Unsplash) (b) An example of a chaotic parking lot (Photo by Ryan Searle on Unsplash)

traffic analysis [5, 6], vehicle tracking [7], etc.

Focusing on the mobility, over the past few decades, the use of cars in large metropolitan areas has massively increased. These trends lead to a significant reduction in the availability of parking lots (see Figure 1.1 (b)). Consequently the amount of time spent driving increases, together with stressful conditions and air pollution. Therefore, smart monitoring of parking stalls aiming to optimize the path from the current driver position to a free parking lot is not only a matter of maximizing profits for the owner or the manager of the parking, but also a matter of public health.

Industrial systems aimed at optimizing the use of parking spaces require methods to count the number of vehicles and free spaces in a parking area. Solutions in this application context can be categorized as follows:

1. *Counter-based* technologies are employed in closed parking areas equipped with a barrier. The controlled access settings allow to detect whenever a car enters or leaves the parking area in order to update the number of vehicles and available parking spaces;

2. *Sensor-based* technologies make use of sensors installed under the asphalt to detect the presence or absence of cars upon it;

3. *Image-based* technologies employ cameras facing the parking area and rely on Computer Vision techniques to count vehicles.

Image-based solutions are economically advantageous over other methods since they do not require specific sensors and can be easily implemented in the context of free-access parking areas. Nevertheless, such approaches have to face several variabilities depending on the positioning of the cameras, the shape of the parking spaces, variable lighting conditions, presence of shadows, occlusions, etc..

The architecture of the system that Park Smart s.r.l. developed follows the Edge Computing design which brings the Computer Vision computation close to the parking area. Their aim is to help private companies and public administrations to manage free entry parking areas, as well closed ones, in order to offer better services to the final customer, i.e. the drivers, and to increase the revenue per stall.

The main problem the company has to face is to find a fast way to deploy working solutions, lowering the labeling effort to the minimum, across different scene, cities, parking areas.

1.1 Parking Management and Computer Vision

The problem of detecting empty vs non-empty parking lots is not new in Computer Vision. Wu et al. [8] proposed a simple pipeline, where patches were extracted and normalized into rectangular patches by using perspective transformation. The color distribution on these patches is computed by the authors and used to feed a Multi-class Support Vector Machine (SVM) for classification purposes. As a final stage, the results of the classification are processed using a Markov Random Field (MRF) to refine potential conflicts between two neighboring patches.

The main objective in [9] was to build an annotated dataset to be exploited by the computer vision community to assess algorithms related to the free parking spaces detection problem. The images were taken in two different parking lots with three different views and different weather conditions (i.e. cloudy, sunny, rainy). In order to perform a benchmark on the dataset, the authors performed different tests

using two different hand-crafted features: Local Binary Patterns [10] and Local Phase Quantization [11]. A Support Vector Machine with a gaussian kernel has been employed for the classification stage. In a first experiment the authors used images sampled from a single parking lot to train and test their system. The main objective of this test was to evaluate the "goodness" of the used features. The second experiment was developed to evaluate the transfer learning property of the approach. Images from a single parking lot have been used as training set, whereas tests where performed with data from the others view/lots. In the third experiment training and testing data were built considering images of all lots and views in order to measure the learning ability of the system.

In [12] a smaller version of AlexNet (named mAlexNet) was adopted to make the detection task executable in real-time on an low-energy embedded device. The authors tested the network developed on the PKLot dataset and on a new dataset, CNRPark-EXT, which is now freely available for the community.

A method based exclusively on image processing techniques was proposed by Yusnita et al. in [13]. The authors mark the real scene painting each stall with a brown circle in the center. In order to decide if a place is available or not the images are thresholded and enhanced using morphological operators. Then the system looks for the circles that are still visible, using an eccentricity based measure to check if the detected blobs are roughly circular. As a last step, the system applies a threshold and counts the remaining spots, giving in output the number of free stalls.

Another approach in literature makes use of trajectories or events to separate empty stalls from non-empty ones. Specifically, Lin et al. [14] employ motion trajectories as feature vectors and then apply the adaptive Gaussian Mixture Model (GMM) and connected component analysis for background modeling and objects tracking.

The work in [15] introduces QuickSpot, a car-driven video analytics solution for on-street vacant parking spot detection designed as a motion detection, object tracking and visual recognition pipeline. To test their performance under different lighting conditions, they created QuickSpotDB, a video database for the vacant parking spot detection problem.

1.2 Aims and Findings

Through an analysis of the domain it was easy to understand that, to build a system robust with high accuracies rates there are several variabilities which have to be considered such as: camera view, shapes of the parking spaces, and other classic variabilities of standard image classification problem such as background, light, deformation, weather. Deep Learning [16] techniques were clearly the approach to use.
We know, through several tests, that we can achieve more than 99.9% of accuracy using binary classification. The time to train different models and to label different datasets for every installation, suggests to find a solution to overpass these limitations. We first tried to decrease labeling effort through Semi-Supervised learning but the solution tried was not suitable and showed to be highly unstable. We tried several methods based on counting cars and parking spots, as solution for fast deployment, which showed some results comparable to image classification and suggest to move the focus on Semantic Segmentation in order to gain a full knowledge of the scene.

1.3 Contribution

The main contributions of this thesis are the following:

- analyze the parking space management related problems through Computer Vision in order to solve fast deployment issues;
- introduce the problem of scene-based domain adaptation for semantic segmentation;
- collect and release a synthetic dataset of images to study the considered domain adaptation problem;
- propose a method to perform scene adaptation using adversarial learning.

The contributions of this thesis have been published and submitted to international journals and conferences:

Conference Di Mauro D, Battiato S, Patanè G, Leotta M, Maio D, Farinella GM. Learning approaches for parking lots classification. In International Conference on Advanced Concepts for Intelligent Vision Systems 2016 Oct 24 (pp. 410-418). Springer, Cham.

Conference Di Mauro D, Moltisanti M, Patanè G, Battiato S, Farinella GM. Park Smart. In Proceedings of Advanced Video and Signal Based Surveillance (AVSS), 2017 14th IEEE International Conference on 2017 Aug 29 (pp. 1-5). IEEE.

Conference Di Mauro D, Furnari A, Patanè G, Battiato S, Farinella GM. A Comparison of Techniques based on Image Classification and Object Detection to Count Cars and Non-Empty Stalls in Parking Spaces. In Proceedings of the 15th International Joint Conference on e-Business and Telecommunications - Volume 2: ICETE, pages 328-336. SciTePress.

Conference Di Mauro D, Furnari A, Patanè G, Battiato S, Farinella GM. Scene Adaptation for Semantic Segmentation using Adversarial Learning. In Proceedings of Advanced Video and Signal Based Surveillance (AVSS), 2018 15th IEEE International Conference on 2018 Nov 29. IEEE.

Journal Di Mauro D, Furnari A, Patanè G, Battiato S, Farinella GM. Scene-Based Domain Adaptation for Semantic Segmentation using Adversarial Learning. Under review In Pattern Recognition.

Journal Di Mauro D, Furnari A, Patanè G, Battiato S, Farinella GM. Estimating the Occupancy Status of Parking Areas by Counting Cars and Non-Empty Stalls. Under review in Journal of Visual Communication and Image Representation

1.4 Thesis Outline

The reminder of this Ph.D. Thesis is divided in 6 chapters all related to problems in the context of scene understanding for parking space management.
Chapter 2 is a presentation of several methods and computer vision tasks which have been used for the work in this thesis. Chapter 3 investigates the binary classification

problem of empty vs non-empty parking space. Chapter 4 describes a first approach to reduce labeling effort using Semi-Supervised Learning. Chapter 5 define two counting problems on the parking space domain tackled with three different methods. Chapter 6 present a Domain Adaptation technique developed in order to solve Scene Adaptation and View Adaptation problems. Finally Chapter 7 concludes the thesis and gives insights for future directions.

Chapter 2

Computer Vision: Background and State of the Art

Deep learning models presented in this chapter are based on the use of Artificial Neural Network. Each level of such network learns a function to transform an input data into a different, usually abstract and compressed, representation.

The word "deep" in "deep learning" refers to the number of layers through which the data is transformed. Neural networks of two layers depth has been shown to be a universal approximator [17] in the sense that it can emulate any function, but the use of extra layers have shown to help learning features in a faster way.

Artificial Neural Network flourished again, after a long stop, thanks to the work in [18]. The work was developed for the ILSVRC-2012 challenge and since that year, deep learning have obtained better results of hand-crafted features on every classical task of computer vision.

2.1 Classification

Image classification refers to the task of assigning a class to an image. It is an example of supervised learning task: the goal is to learn a function that maps an image (input) to a class (output). In general, in supervised learning tasks, each example is a pair consisting of an input object and a desired output value. An optimal model will allow to correctly determine the class labels for unseen instances, in this case we talk about generalization.

In the following sections we introduce five deep learning architectures (LeNet, AlexNet, GooLeNet, ResNet and VGG) which have been used to produce the work presented in this thesis.

2.1.1 Gradient-based learning applied to document recognition (LeNet)

In recent years, Neural Networks have re-gained attentions in computer vision. In particular nowadays we talk about **Convolutional Neural Networks (CNN)**. CNNs architectures make the explicit assumption that the inputs are images and allows to encode certain properties to be considered in the context of visual understanding into the Neural Network architecture, enabling also an efficient forward function implementation and reducing the amount of free parameters in the network. Such approach found their roots in the work by LeCun et al. [19].

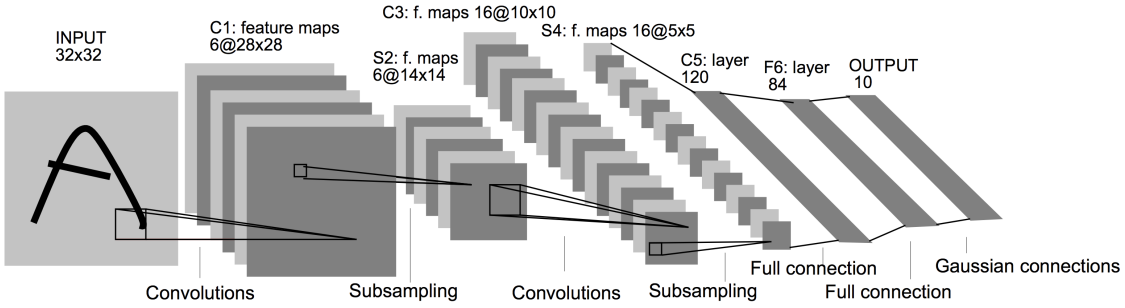


Figure 2.1: An illustration of the LeNet architecture (Image from [19])

In this network we find some core building block of a Convolutional Neural Network: Convolutional layer, Spatial Pooling and Fully Connected Layer.

The Convolutional layer's parameters consist of a set of learnable filters, every filter is small along width and height, but extends through the full depth of the input volume. During the forward pass, we convolve each filter across the width and height of the input volume and compute dot products between the entries of the filter and the input at any position. The result of such operation will produce a 2-dimensional activation map that gives the responses of that filter at every position. The network will learn filters that activate when they encounter a visual feature useful for the task. Each Convolutional layer is composed by a set of filters, and

each of them will produce a separate 2-dimensional activation map, which will be stacked along the depth producing the output volume.

Spatial Pooling (also called subsampling or downsampling) is another core layer in modern CNNs. The goal is to reduces the dimensionality of each feature map but retains the most important information. Spatial Pooling can be of different types: max, average, sum, etc., depending on the function used to filter the most important information, in case of Max Pooling, we define a spatial neighborhood (e.g., a 2×2 window) and take the largest element from the activation map within that window.

Finally the Fully Connected layer is a traditional Multi Layer Perceptron that uses a softmax activation function in the output layer. "Fully Connected" implies that every neuron in the previous layer is connected to every neuron on the next layer.

2.1.2 ImageNet Classification with Deep Convolutional Neural Networks (AlexNet)

In [18] was presented a CNN for the ILSVRC-2012 competition. The Convolutional Neural Networks proposed is known as **AlexNet** and in 2012 won the competition with a good margin of accuracy respect to other methods. AlexNet contains eight layers: the first five are convolutional layers and the remaining three are fully-connected. The last fully-connected layer is a 1000-way softmax which produces a distribution over 1000 object classes.

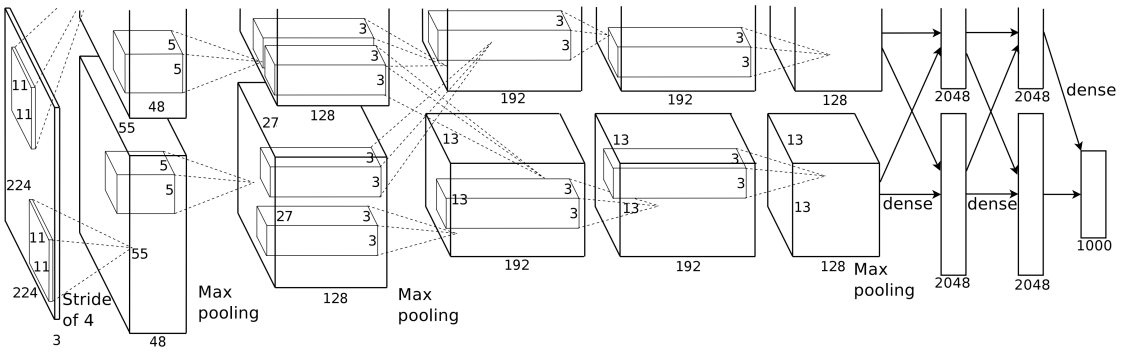


Figure 2.2: An illustration of the Alexnet architecture (Image from [18])

The Network had an architecture close to LeNet, but it was deeper, bigger, and it featured Convolutional Layers stacked on top of each other, previously it was common to only have a single Convolutional layer always immediately followed by a Spatial Pooling layer. It uses Rectified Linear Unit (ReLU) [20] instead of the hyperbolic tangent function (Tanh) to add non-linearity, such function accelerates the speed of the learning process maintaining the same accuracy, finally it use Dropout [21] instead of regularization to deal with overfitting.

2.1.3 Going Deeper with Convolutions (GoogLeNet)

A further architecture which was introduced by Szegedy et al. [22] is GoogLeNet. Such model won the ILSVRC-2014 challenge with a TOP-5 test accuracy of 93.3%. This architecture is composed by 22 layers and a newly introduced building block called inception module. This new approach proved that CNN layers could be stacked in more ways than a typical sequential manner. We talk of a Network in Network (NiN) layer composed by a pooling operation, a large-sized convolution layer, and small-sized convolution layer computed in parallel and followed by 1×1 convolution operations to reduce dimensionality. Thanks to those modules, this network reduces the number of parameters and operations being efficient on memory usage and on computational cost. In Figure 2.3 (a) is depicted the architecture of the network.

2.1.4 Very Deep Convolutional Networks for Large-Scale Image Recognition (VGG)

With the name of VGG we indicate a CNN architecture introduced by the Visual Geometry Group (VGG) from the University of Oxford in [23]. They proposed various models and configurations of deep CNNs, one of them was submitted to the ImageNet Large Scale Visual Recognition Challenge (ILSVRC-2013). That model, known as VGG-16 or VGG-19 due to the fact that it is composed by 16 or 19 weight layers, became popular thanks to its achievement of 92.7% TOP-5 test accuracy. The main difference between VGG models and its predecessors is the use of a stack of convolution layers with small receptive fields in the first layers instead of few layers with big receptive fields leading to less parameters and more nonlinearities resulting

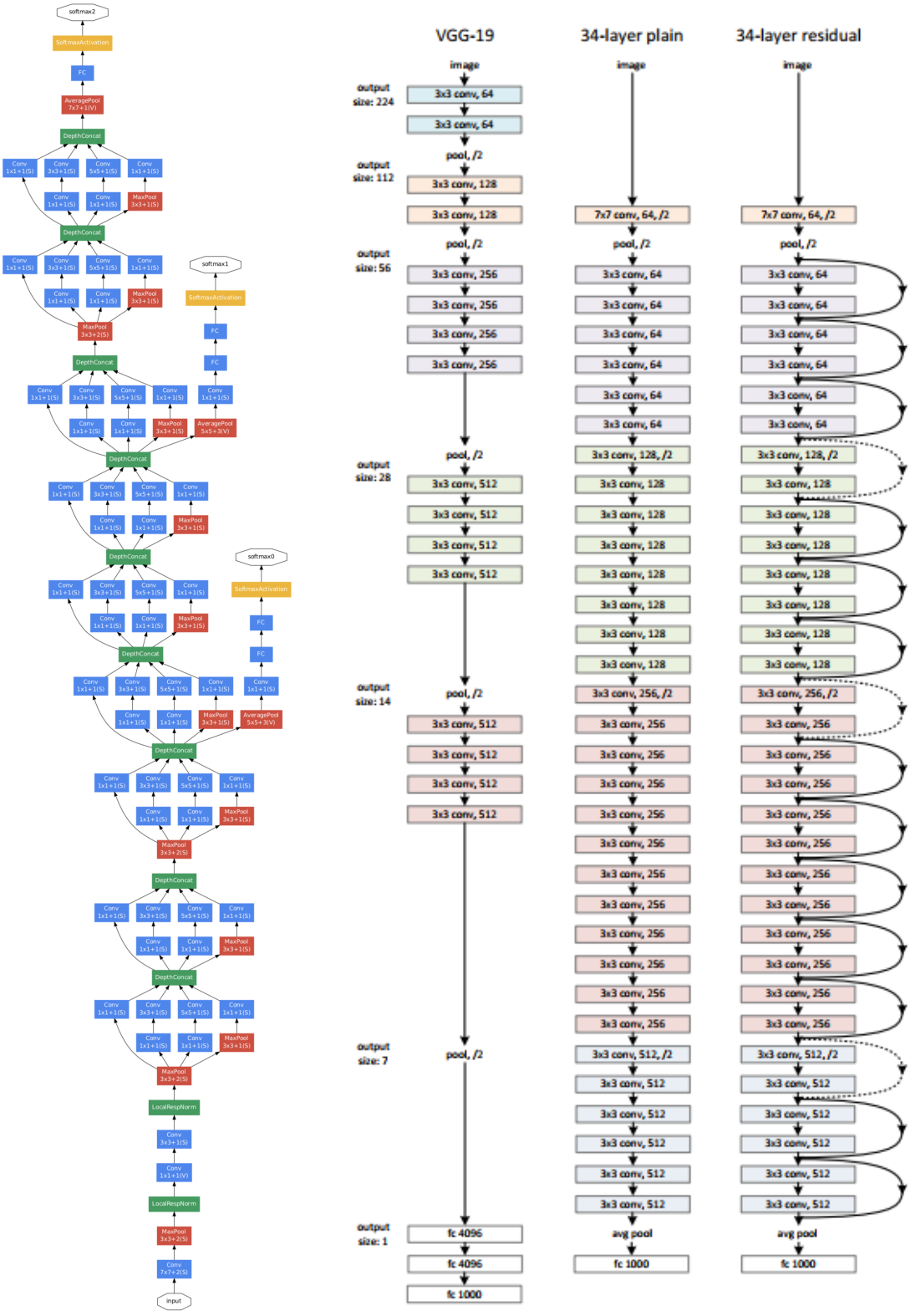
in a more discriminative function and an easier model to train. An example of the model with 19 weight layers is depicted in Figure 2.3 (b). A downside of VGG is its computational cost: it uses a lot more memory and parameters, most of it depends from the first Fully Connected layer. Since then it was discovered that Fully Connected layers can be removed with no performance decay, significantly reducing the number of parameters.

2.1.5 Deep Residual Learning for Image Recognition (Resnet)

As we saw in the previous paragraphs a strong trend has been to make the networks architectures deeper, with VGG using 19 layers and GoogLeNet 22, and it has been noted that making layers wider by adding more nodes it is not to be preferred since increase overfitting. Adding more layers is better, but because of the vanishing gradient problem, model weights of the first layers can not be updated correctly through the back propagation of the error gradient: the chain rule multiplies error gradient values lower than one and when the gradient error comes to the first layers, its value goes to zero. That is the objective of ResNet [24]: it preserves the gradient thanks to the identity matrix. Given a shallower network we can add extra layers and make it deeper without losing accuracy or increasing error using identity mappings, they become equivalent to the shallower network. They should produce no higher training error than its shallower counterpart. The authors call this approach a solution by construction. In Figure 2.3 (b) we can see a 34 layer ResNet compared with VGG-19.
ResNet shows us that you should use as big of a neural network as your computational budget allows, without fearing to overfit it, and use other regularization techniques to control overfitting.

2.2 Object Detection

Object detection is the computer vision task that deals with detecting instances of semantic objects of a finite set of classes in images and videos. Ground Truth labels are bounding boxes around the objects, in other words four corners of the smallest rectangle which contain all the object. It was introduced in [25], all the state of the



(a) (b)

Figure 2.3: (a) An illustration of the GooLeNet architecture (Image from [22]) (b) A 34 layer ResNet with VGG-19 side by side (Image from [24])

art approaches for object recognition, at the time, reduced the problem to a present vs not-present task.

In the following sections we introduce two state-of-the-art methods, Faster R-CNN, which has been used in this thesis for work presented in Chapter 5, and YoLo.

2.2.1 Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks (Faster RCNN)

Faster R-CNN [26] is one of the state-of-the-art deep learning architecture for object detection. The input images are represented as tensors which are passed through a pre-trained CNN until an intermediate layer, ending with a convolutional feature map, it uses this as a feature extractor for the next part.

The following part is called Region Proposal Network (RPN). Using the features that the CNN computed, it is used to regress to a predefined number of regions (bounding boxes), which may contain objects.

The hardest issue with using Deep Learning for object detection is generating a variable-length list of bounding boxes. The problem is solved in the RPN by using anchors: fixed sized reference bounding boxes which are placed uniformly throughout the original image. Instead of having to detect where objects are, the problem is modeled in two parts.

Using the features extracted by the CNN and the bounding boxes with relevant objects, through applying Region of Interest (RoI) Pooling it extracts those features which would correspond to the relevant objects into a new tensor.

The R-CNN module uses that information to classify the content in the bounding box or discard it and adjust the bounding box coordinates to better fits the object. In Figure 2.4 the architecture of Region Proposal Network (RPN) and some example detections using RPN proposals on PASCAL VOC 2007 test.

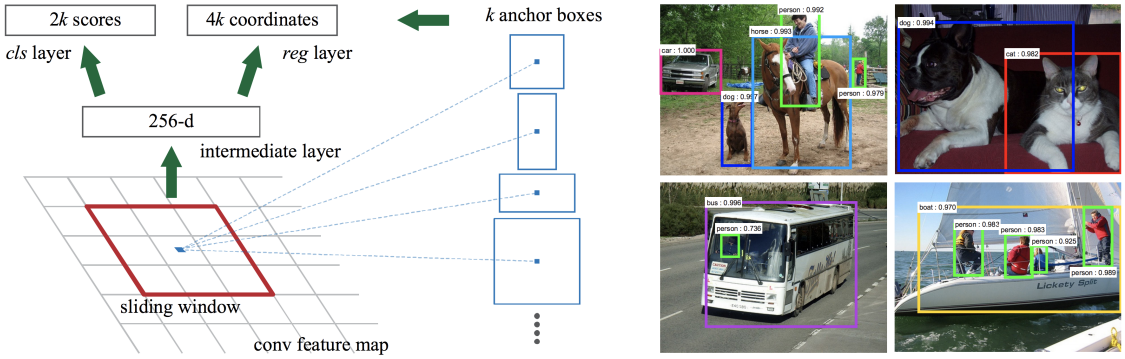


Figure 2.4: Region Proposal Network (RPN) and some example detections using RPN proposals on PASCAL VOC 2007 test. (Image from [26])

2.2.2 You Only Look Once: Unified, Real-Time Object Detection (YOLO)

Ususally, as we saw with Faster R-CNN, detection algorithm based on deep learning techniques repurpose classifiers architectures to perform detection. They apply the model to an image at multiple locations and scales and, depending on the algorithm, considers detections part of the image which respond with higher scores.

The work by Redmon et al [27] uses a totally different approach. They pass a full image to a single neural network, which divides the image into regions predicting bounding boxes and probabilities for each region. These bounding boxes are weighted by the predicted probabilities.

Their model has several advantages, it looks at the whole image at test time such as its predictions are informed by global context, all predictions are done with a single network evaluation unlike systems like R-CNN which require thousands resulting in an extremely fast network.

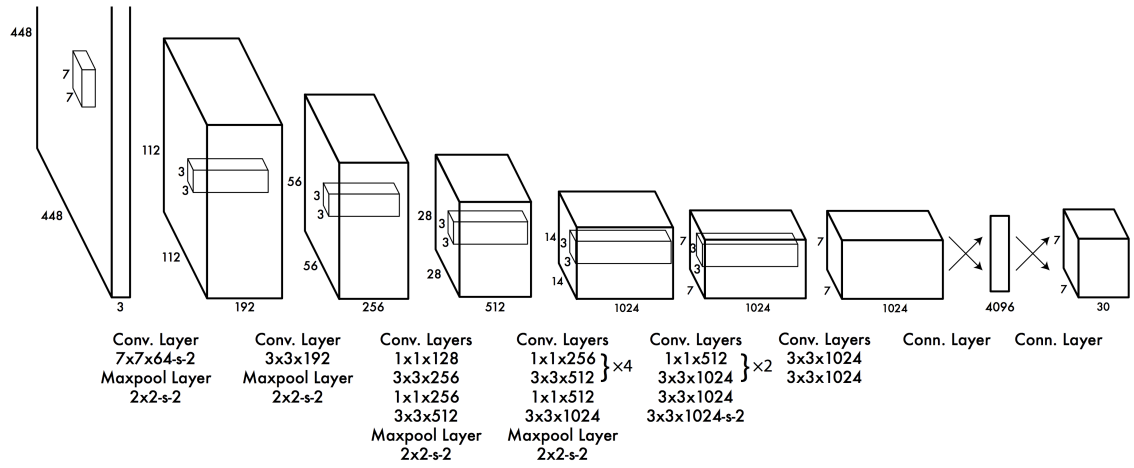


Figure 2.5: Architecture of YOLO network (Image from [27])

2.3 Semantic Segmentation

Semantic segmentation is the computer vision task in which specific regions of an image are labeled according to the semantic meaning of the pictured object, first works of this task appear in [28, 29]. This task is also referred as dense prediction, because we are predicting the class of each pixel in the image. In Semantic Segmentation task the numerosity of objects is not relevant, if you have several objects of the same category in your input, the output map does not distinguish these as separate objects.

In segmentation each pixel contains a class label represented as an integer. In the following sections we introduce four state-of-the-art methods, two of them (SegNet and PSPnet) have been used in this thesis for work presented in Chapter 5 and Chapter 6.

2.3.1 SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation

Badrinarayanan et al. [30] presented SegNet. The network is composed by a stack of encoders followed by a corresponding stack of decoders. The up-sampling is performed by the decoder using the max-pool location indexes generated by the encoder. The final decoder output feature maps are fed to a soft-max classifier for

pixel-wise classification. In Figure 2.6 the SegNet architecture is depicted, it shows how pooling indexes are used to drive upsampling.

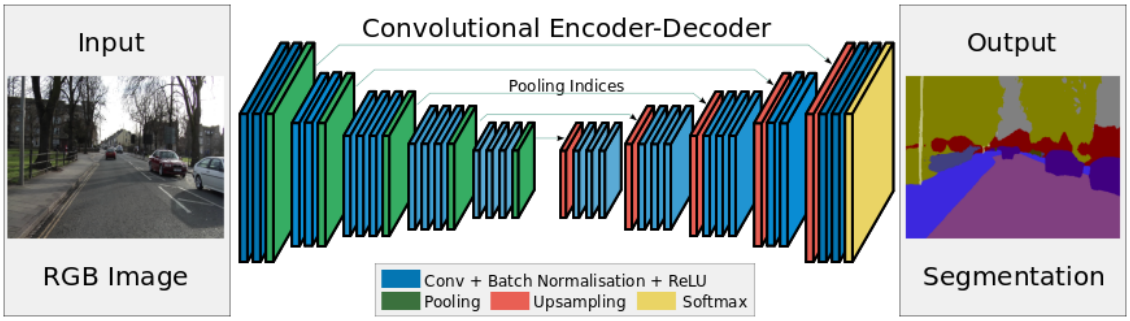


Figure 2.6: An illustration of the SegNet architecture (Image from [30])

2.3.2 Fully Convolutional Networks for Semantic Segmentation (FCN)

Previous works investigated the use of Deep Learning to address the problem of semantic image segmentation. Long et al. [31] proposed to exploit existing CNNs for classification to produce dense image segmentation masks. Classification networks are modified by substituting fully connected layers with convolutions. The low resolution feature map produced by the network is hence up-sampled using dilated de-convolutions to match the resolution of the input image. The resulting architecture is shown in Figure 2.7.

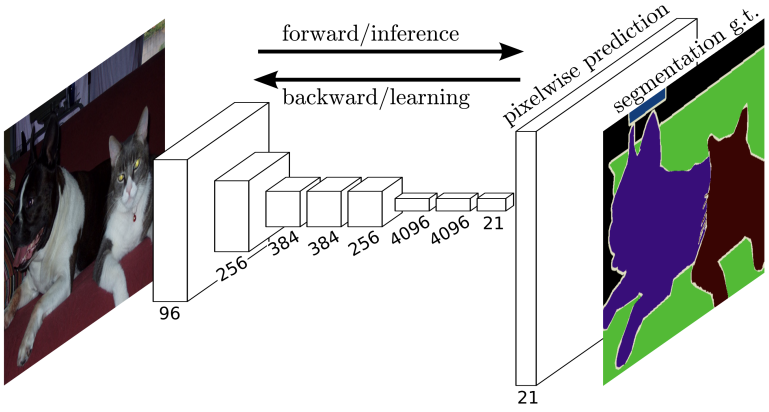


Figure 2.7: An illustration of the FCN architecture (Image from [31])

2.3.3 DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs

Another work which addresses the task of semantic image segmentation is [32]. The first main contribution of this work are "atrous convolution" which allows to explicitly control the resolution at which feature responses are computed and also to effectively enlarge the field of view without a corresponding increasing in the number of parameters or the amount of computation. A second contribution is the use of atrous spatial pyramid pooling (ASPP) to segment objects at multiple scales, thus capturing objects and image context at multiple scales. Finally they improve localization of object boundaries through the use of graphical model, the final layer is connected with a fully connected Conditional Random Field (CRF). In Figure 2.8 an illustration of the process of DeepLab approach.

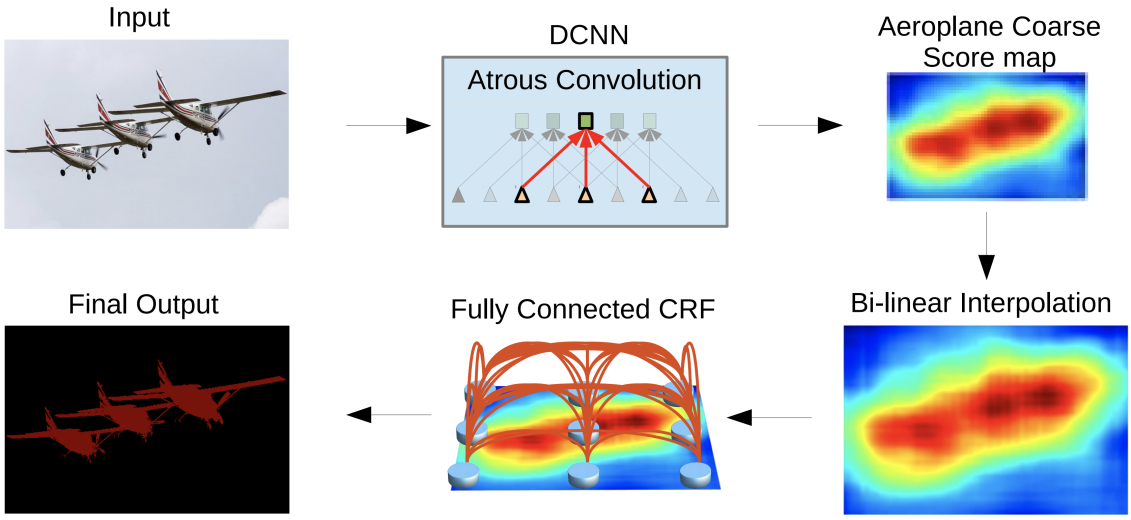


Figure 2.8: An illustration of the DeepLab method (Image from [32])

2.3.4 Pyramid Scene Parsing Network (PSPnet)

Zhao et al. [33] proposed to exploit global contextual information by context aggregation through a pyramid pooling module. The considered global prior representation is effective to produce good quality results on the scene parsing task. The considered approach achieves state-of-the-art performance on various datasets. The

architecture, and the main contribution, the pyramid pooling module is shown in Figure 2.9.

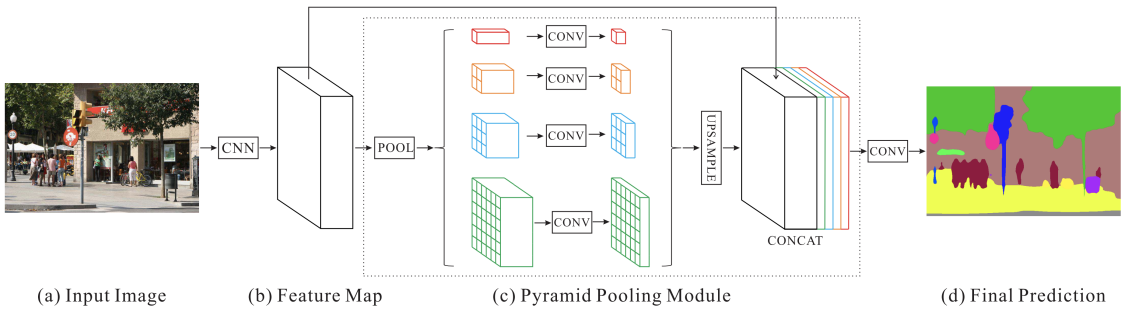


Figure 2.9: An illustration of the PSPnet architecture (Image from [33])

2.4 Generative Models and Applications

Generative models represent probability distributions. There are several Generative Models, among the others we can cite Boltzmann Machines [34], Variational Autoencoders (VAE) [35], Fully visible belief networks [36], etc. The main difference between each model is how they calculate such distribution. Some evaluate probability function explicitly, while others support operations that implicitly require knowledge of it, such as drawing samples from the distribution. In particular Generative Adversarial Networks, which will be shortly described in the following section, belongs to this second family.

2.4.1 Generative Adversarial Networks

Since their introduction in 2014 [37], Generative Adversarial Networks (GANs) have been employed in a series of applications, including super resolution [38], next video frame prediction [39], and generative visual manipulation [40].
In a GAN setup, two differentiable functions, represented by neural networks, are locked in a game. The two players which are called the generator and the discriminator have different roles in this framework.
The generator tries to produce data that come from some probability distribution, while on the other side the discriminator acts like a judge. It gets to decide if the

input comes from the generator or from the true training set. We are in a situation similar to a police officer (the discriminator) looking for fake money and a counterfeiter (the generator) that's printing fake money. It is a two-player zero-sum game where generator and discriminator try to find a balance between them.

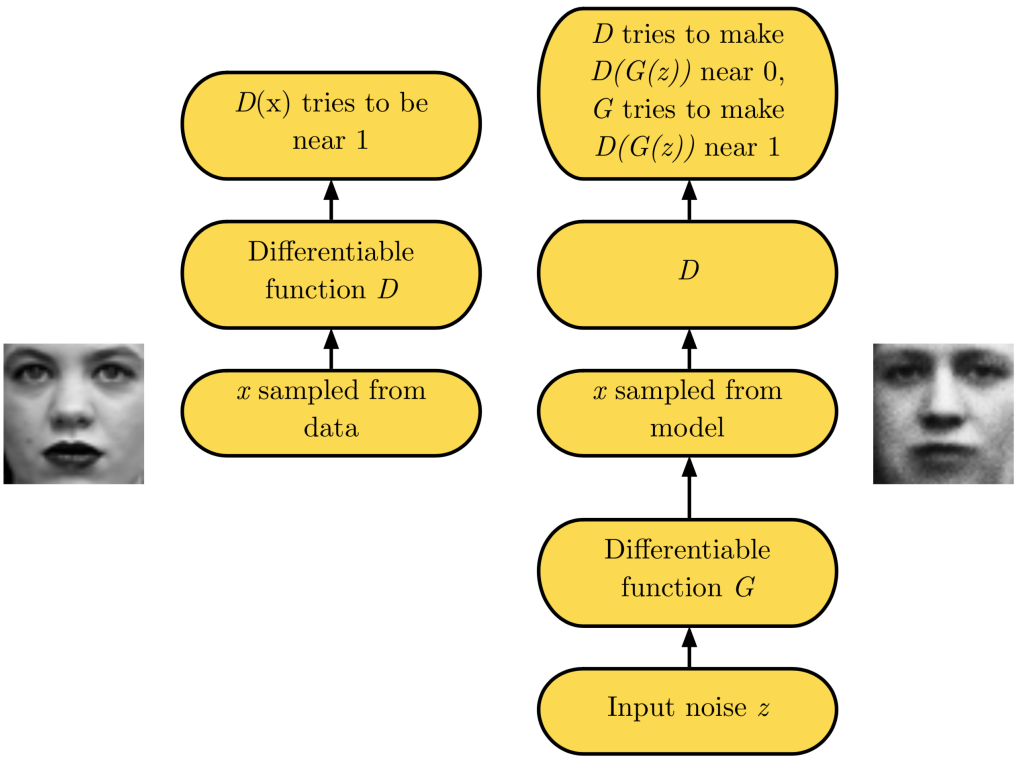


Figure 2.10: General GAN framework (Image from [41])

In Figure 2.10 an example of application GAN for face generation.

2.4.2 Image Translation

In particular, our work is related to the work of Isola et al. [42], who addressed the problem of image-translation through the use of Generative Adversarial Networks is introduced. The formulation of the image-translation problem assumes two different domains X and Y. The goal of image-translation is to model a function $F : X \rightarrow Y$ which transforms an input $x \in X$ to an output $\hat{y}$ which is indistinguishable from elements which could be sampled from Y. The work has further been extended to

the case of unpaired images in [43]. In this case, given images from two domains X and Y, two translation functions are modeled: $F : X \rightarrow Y$ and $G : Y \rightarrow X$. The mappings are learned using unpaired samples by enforcing a cycle consistency loss.

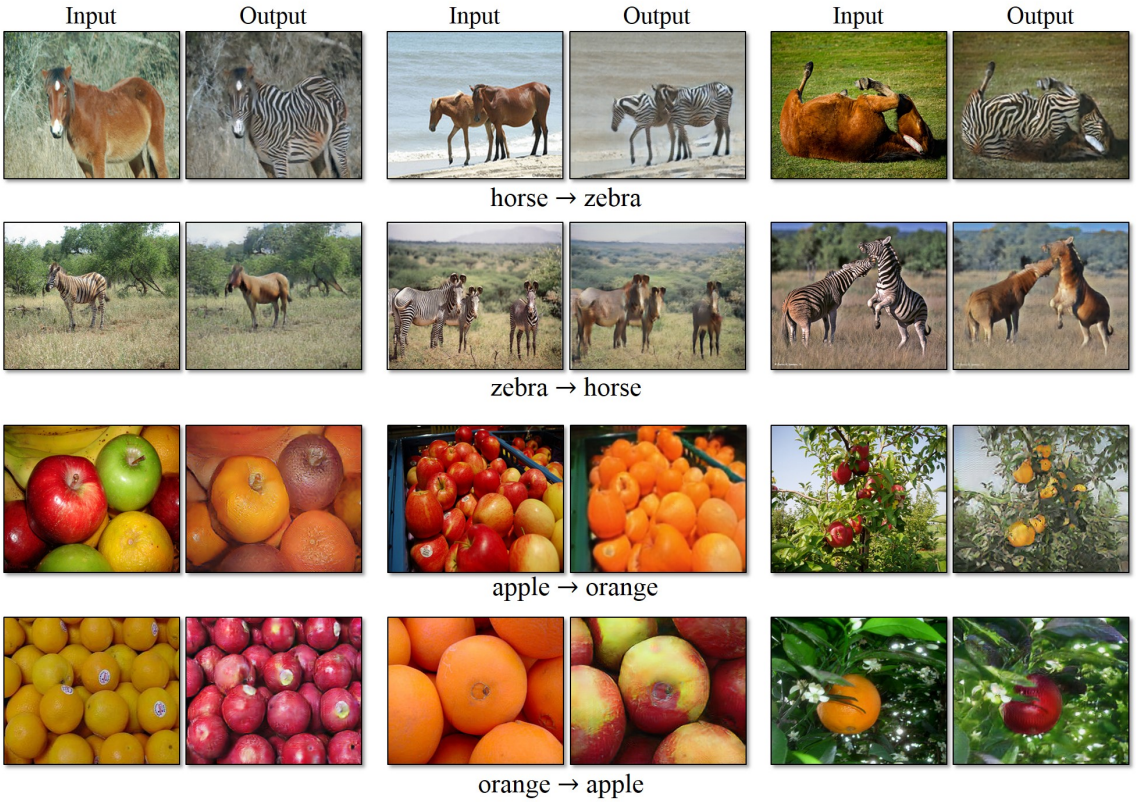


Figure 2.11: An example of CycleGAN Object Transfiguration (Image from [43])

In Figure 2.11 an example of application of Image Translation through CycleGAN.

2.4.3 Transfer Learning and Domain Adaptation

All the networks presented so far can be applied to problems different to the original one performing a fine-tuning step. The fine-tuning method is a transfer learning mechanism. Usually researchers do not train the Convolutional Network from scratch but, most of the time, it is common to use a pre-trained Convolutional Neural Networks on a very large dataset (e.g., Alexnet over ImageNet) and only fine-tune (recalculate weights) some, or the very last, higher-level portion of the

network. The rationale behind this choice comes from the observation that the earlier layers, because of their convolutional nature, extract low visual features that are useful to many general computer vision tasks. With fine-tuning the higher layers are trained for the specific task. Domain Adaptation is a particular case of transfer learning where the Source Domain is a labeled set of data, while the Target Domain is not [44]. The main difference between canonical Supervised Learning and Domain Adaptation is, indeed, that in the latter situation we study two different, but related, distributions.

2.4.4 Domain Adaptation for Semantic Segmentation

Domain adaptation for semantic segmentation is gaining attention in the last couple of years, a first work, to the best of our knowledge, is Hoffman et al. [45] which proposed to perform global domain alignment using a novel semantic segmentation network with fully convolutional domain adversarial learning. Category specific adaptation is performed through a generalization of constrained weak learning, with explicit transfer of the spatial layout from the source to the target domain. In a subsequent work, Hoffman et al. [46] proposed to adapt representations both at the pixel-level and at the feature-level through cycle-consistency without requiring aligned pairs.
The model has been applied to a variety of visual recognition and prediction settings, including semantic segmentation of road scenes. Sankaranarayanan et al. [47] addressed the problem of domain adaptation for semantic segmentation with a method composed by an embedding network, a pixel-wise classifier, a generator network and a discriminator network. In [48], Tsai et al. employed a multi-level adversarial network to perform output space domain adaptation at different feature levels. The algorithm consists of two modules: a segmentation network G and the discriminator D_i, where i indicates the level of a discriminator in the multilevel adversarial learning.

Chapter 3

Empty vs Not-empty Parking Space Classification

Classification is the task where the computer vision community has obtained great results since the introduction of deep CNNs. Thus we decided to tackle the problem to decide if a parking space is empty or not as a classification task over patches corresponding to parking lots. This approach is well suited for the most cases. The main idea is to divide each frame captured by the camera in several crops, where every crop is a square image corresponding to a parking space.

3.1 Park Smart: The Overall System at Glance

Despite the problem of detecting empty vs non-empty parking lots is "simple" it can be easily intractable. Lets think about a single camera which streams every frame to a centralized server. Then multiply the band needed by one stream for the several cameras, hundreds, or thousands, or even millions, needed to monitor several smart cities areas. Note that a mid-sized city could need a number between 800 and 1000 cameras to monitor all the parking spaces.

It is clearly infeasible, or at least really expensive and not economically profitable to manage such kind of streaming traffic in real-time. We thus decided for a more scalable approach by bringing the computation close to the camera which acquire the stream using dedicated embedded systems that will send the results to the main server system.

Our architecture is described by Fig. 3.1. It has four main components:

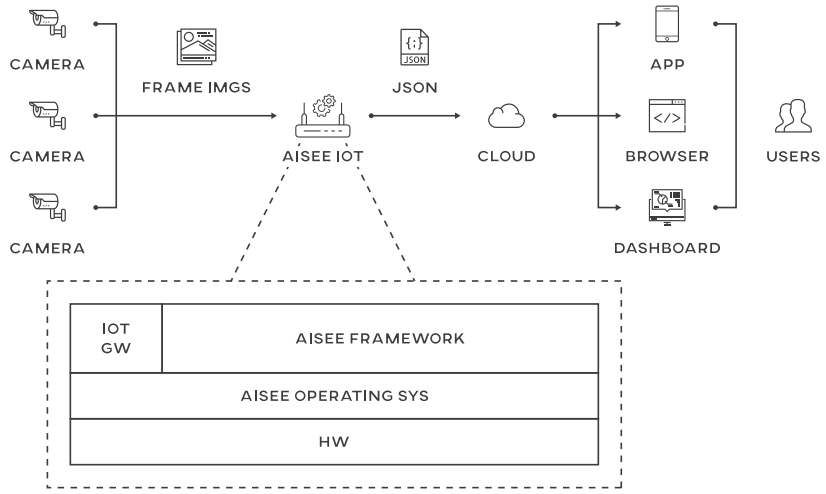


Figure 3.1: This diagram shows the current Park Smart system: images and videos are captured by cameras which send them to the AISEE embedded where the computation is performed. From there the information about the parking status is send to the cloud in order to be viewed by users.

Cameras We use wide angle cameras to optimize the number of parking spaces monitored. Our approach is not vendor locked. To have best results the resolution needed is at least 50px per side for each parking space.

AISEE IoT We analyze the video stream as closest as possible to the camera. It is an embedded system capable of elevated computing power, enough to do inference using deep learning models. Once inference is done the results are sent to the cloud platform. The embedded operating system has been developed with security, privacy and resilience in mind. We can deploy several AISEE IoT boxes depending on the number of cameras and the dimension of the installation.

Cloud We collect all the information from several installed embedded systems through a cloud platform which is scalable by design.

Presentation layer The system is accessible through different kind of appliances:

- The **dashboard** is the business and administration front-end which allows all the operations and to manage the installations (e.g. to add new

cameras, configure cameras, add embedded, remove embedded and upgrade them, etc.).

- The **mobile app** or **browser** are the ending point for the people who are looking for a free spot where to park.

3.2 Classification of parking stalls

To investigate the approach, before producing our dataset, we used PKLot dataset [49], it has 12417 images with resolution of 1280×720 pixels. This dataset is really interesting for us for the following key features

- images were taken from three different parking areas;
- cameras were positioned at different heights;
- images have strong variability: such as the presence of shadows, over-exposition, low light, difference in perspective.

We sampled three datasets, one for each parking area, and fine-tuned AlexNet. The results are reported in Table 3.1.

Table 3.1: Results using a fine-tuned AlexNet on PKLot

Sample	**Train**	**Val**	**Test**	**Accuracy**
UFPR05	19281	4820	24101	99.93%
UFPR04	20000	5000	25000	99,96%
PUC	20000	5000	25000	99,92%

We tested the system on other three dataset considering images of a parking area composed by 46 parking spaces. The images were acquired in Catania, Sicily, during summer, autumn and winter 2015 in order to have as much variabilities as possible (light, weather, different cars, etc) and at different time of the day. To cover the parking space area the images have been acquired from eight cameras with Full-HD resolution extracted from motion jpeg streams. The sampled images have been cropped to extract stalls and manually labeled. Specifically each crop has been assigned a free or occupied label.

Table 3.2: Results obtained considering different CNN models and three dataset. In particular, DS1 has 17688 train images, 3924 in val and 21612 in test; DS2 has 20636 train images, 4578 in val and 31374 in test; DS3 has 13032 train images, 2820 in val and 25212 in test

CNN Models	DS1	DS2	DS3	Avg. Accuracy	Footprint
AlexNet [18]	98,80%	99.20%	93.82%	97,27%	217M
GoogLeNet [22]	99.72%	99.58%	99.26%	99.52%	40M
VGG16 [23]	99.13%	98.70%	94.91%	97.58%	528M

We analyzed different methods of cropping images, but, in most cases, the methods did not have an impact on the final classification results.

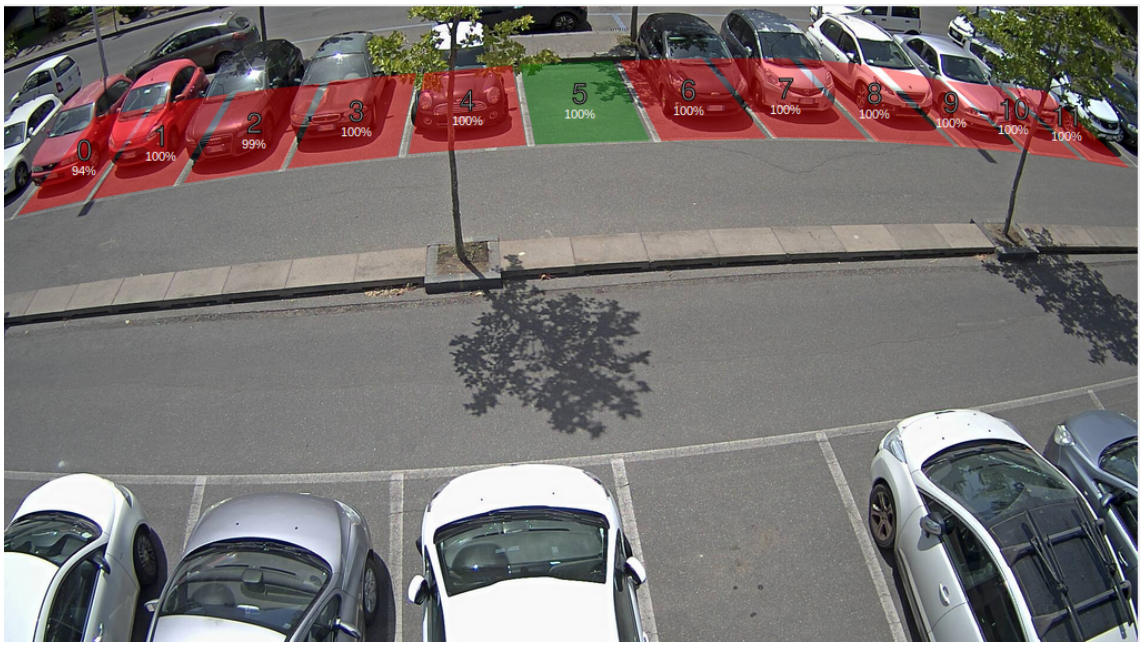


Figure 3.2: An example of classification of one camera. Best viewed in colors.

To perform our experiments we used the Caffe library [50] taking advantage of GPU optimized code. To fine-tune the networks we used a machine equipped with four **NVIDIA GeForce TITAN X with 12Gb of DDR5 RAM**.

In order to find the best solution, balancing accuracy, classification speed and model footprint, we have investigated different models known in literature. The results are reported in Table 3.2. As we can see all the different models work quite well, with accuracy of 97% or more. GoogLenet (without the fully-connected layer)

Figure 3.3: Here there are some misclassified images of parking spaces. Best viewed in colors.

is the slowest to train but is the faster at inference time and the one with the smallest footprint, while VGG16 and AlexNet weigh far more.

In Figure 3.2 we show an example of classification. The camera is monitoring 12 parking spaces. On every parking space, an overlay displaying the confidence of belonging to empty or non empty class is shown. In Figure 3.3 we show some misclassification examples. Most of the errors depends from high occlusions and unconventional geometries. To better asses the results, a video demonstrating the proposed solution is available at the following url: `http://iplab.dmi.unict.it/ParkSmart/`

3.3 Discussions

In this chapter we have presented the Park Smart system, an end-to-end pipeline for smart parking assistance and management. The infrastructure makes use of an IoT device which allows to perform the computation on the "borders" of cloud, implementing the Edge Computing paradigm. Moreover, we have discussed a classification based computer vision algorithm able to classify parking spaces, given their spatial configuration. The work developed in this chapter is a baseline for Semi-Supervised approaches presented in 4.

Chapter 4

Semisupervised Learning Approach Using Pseudolabels

The approach presented in Chapter 3 used supervised learning algorithms to deal with the problem of classifying a parking space as empty or non-empty. The main drawback of this kind of approaches is the construction of a big dataset to train the classifier. Considering the needs of a company deploying an image-based parking monitoring system, the effort of manually labeling the images to build such big training dataset can became problematic since it is time consuming (i.e., it has high costs for the company). Moreover, it is difficult to build in advance a proper dataset for training by considering all the possible variabilities which can be observed in different parking spaces. Variabilities depend on the positioning of the cameras, shapes of the parking lots, as well as from the classic variabilities which can be considered in a standard image classification problem (e.g. appearence of vehicles, lighting conditions, presence of shadows, occlusions, etc. See Fig. 4.1). Moreover, most of the time the images to build the training set (and hence to include possible variabilities in the learning process) are only available after the deployment of the parking monitoring system. It is hence important to reduce the labeling time adopting learning methods able to propagate the previous learned model (i.e. models already working on others parking lots) to the new parking lot to be monitored.

The above considerations have motivated the study presented in this chapter. In Supervised learning, the labeled datasets are fundamental for the training, but are "difficult" to create. On the other side, it is simple to create partly labeled sets. The main goal is how to propagate the labels from few labeled samples to the full dataset composed by both labeled and unlabeled data. Semi-supervised learning is

usually proposed to deal with this kind of problem. In order to exploit unlabeled data, Semi-supervised approaches make some assumptions to the underlying data distribution. In particular, three assumptions are the most used in literature [51]:

- **The Smoothness assumption:** if two feature points x_1, x_2 in a high-density region are close, then they should have close corresponding outputs y_1, y_2.
- **The Cluster assumption:** if feature points are in the same cluster, they are likely to be of the same class.
- **The Manifold assumption:** high-dimensional data likely lie roughly on a low-dimensional manifold.

We evaluate the performances of a semi-supervised learning approach in comparison with respect to a classic supervised learning one in order to benchmark the problem of parking space monitoring. Specifically, considering the high interest of the community on the deep learning architectures, we compare the well-known supervised learning CNN architecture called Alexnet [18] with respect to the recent pseudo-label approach for semi-supervised learning of deep neural networks [52]. To perform this comparison we consider two different datasets; the recent introduced PKLot dataset [9] and a new dataset called Parksmart dataset (PSD) whose images have been collected in real scenarios by considering different variabilities.

Experiments have been done to evaluate the performances of Alexnet [18] and pseudo-label [52] with respect to different training settings. Results show that for the problem addressed (empty vs. non-empty space) Alexnet outperforms pseudo-label in all cases and independently from the number of training samples used as input. The lesson learned is that for the considered problem a supervised learning technique is to be preferred also in cases where only few labeled samples are available as training.

4.1 Methods

To compare supervised vs semi-supervised methods we have exploited recent approaches presented in literature [18] [52]. In the following we summarize the employed methods.

Figure 4.1: Example of parking space variability in geometry, luminance, occlusions, shapes

4.1.1 Deep Convolutional Neural Networks and Fine-Tuning

As a baseline we use AlexNet vanilla implementation which was explained in 2.1.2. In our case we change the last fully-connected layer from a 1000-way to a 2-way softmax which produces a distribution over 2 classes.

4.1.2 Pseudo-Label: The Simple and Efficient Semi-Supervised Learning Method for Deep Neural Networks

In [52] is proposed a network trained in a semi-supervised fashion. The training is achieved using at the same time labeled and unlabeled data. To the unlabeled data is assigned the label that the network computed on the forward pass (this is why it is called "Pseudo Label"). During the training process the loss function calculated on labeled data and pseudo-labeled ones are summed using the following formula:

$$L = \frac{1}{n}\sum_{m=1}^{n}\sum_{i=1}^{C} L(y_i^m, f_i^m) + \alpha(t)\frac{1}{\tilde{n}}\sum_{m=1}^{\tilde{n}}\sum_{i=1}^{C} L(\tilde{y}_i^m, \tilde{f}_i^m) \tag{4.1}$$

where n is the number of labeled data, $\tilde{n}$ the number of unlabeled data, C the classes, t is the number of iterations, y and f are the labels and network result for labeled data, $\tilde{y}$ $\tilde{f}$ are pseudo-labels and network result for unlabeled data and $\alpha(t)$ is defined as

$$\alpha(t) = \begin{cases} 0 & \text{if } t < T_1 \\ 1 & \text{if } T_1 \leq t < T_2 \\ a_f & \text{if } T_2 \leq t \end{cases} \tag{4.2}$$

where $a_f = 3$ and $T_1 = 100$, $T_2 = 600$.

4.2 Dataset

In our experiments we used data sampled from the PKLot dataset [9] and a new dataset which is referred with the acronym PSD. Details on the dataset are given in the following.

4.2.1 PKLot Dataset

The dataset consists of 12,417 images resulting in 695,899 samples of parking spaces which have been manually labeled by the authors of [9]. The images were taken within a 5 minutes time interval, during a 30 days period, using a Full-HD camera (Microsoft LifeCam HD-5000) with resolution of 1280×720 pixels. The cameras were positioned at very high altitude to minimize occlusions (see Fig. 4.2). The images were stored in high quality JPEG (100% quality). To crop and to label every space in the images as free or occupied, the authors of [9] developed a specific tool able to save information of each space in an Extensible Markup Language file (XML). To remove rotation variability every parking space patch has been rotated horizontally or vertically depending of their original rotation angle. The key features of this dataset are the following:

- images were taken with different weather conditions (sunny, rainy and overcast periods);
- images were taken from different parking lots;
- cameras were positioned at different heights;
- images have strong variability: such as the presence of shadows, low light, over-exposition, difference in perspective.

4.2.2 PSD Dataset

The proposed PSD dataset was aquired from August 2015 to November 2015 in a parking lot of the University of Catania. In order to capture variabilities we have considered three time slot during the day:

1. from 06:00am to 09:00am (3 hours)

Figure 4.2: Examples of images belonging to PKlot dataset

2. from 12:30pm to 02:30pm (2 hours)

3. from 05:30pm to 09:00pm (3.5 hours)

The monitored parking lot is composed by 46 parking spaces. To cover the whole parking lot the data have been acquired by four cameras with a resolution of 1920×1080 extracted from motion jpeg registration. The sampled images have been manually labeled. Specifically, for each image the different parking spaces have been manually labeled as free or occupied. For experimental purpose the final set of parking spaces is composed by 270796 crops. Examples of PSD parking lots are reported in Fig. 4.3.

Figure 4.3: Example of images belonging to PSD dataset

Table 4.1: Considered dataset and training images

		Training			
Dataset	**Images**	**0.17%**	**1%**	**1.7%**	**5%**
PKLot	72000	120	720	1200	3600
PSD	144000	240	1440	2400	7200
PSD*	42000	70	420	700	2100

4.3 Experiments and Results

To perform our experiments we used Caffe library [50]. All the code was developed using the Python api. We took advantage of GPU optimized code running the tests on a machine equipped with four NVIDIA GeForce TITAN X with 12Gb of DDR5 RAM.

To test how the fine-tuned AlexNet and the pseudo-labeling method behave we have performed different experiments varying the number of training images. Following the rationale in [52], as training we selected a subset of images from PKLot and from PSD, equal to 0.16%, 1%, 1.6%, 5% of the entire dataset (see Table 4.1).

The network described in [52] is a neural network with 1 hidden layer. Rectified

Linear Unit is used for hidden units, Sigmoid Unit is used for output units. The number of hidden units is 5000. Pseudo-label, is the method to feed a neural network with labeled and unlabeled data at the same time, thus we decided to implement such method using an augmented Alexnet able to behave in such way to proper compare the results. In particular we added an input layer and a concatenation layer on the front of the network to have two different entrances for labeled and unlabeled data and to concatenate them before feeding the network. At the end of the network, we added a slicing layer and a second loss layer in order to calculate different loss values considering the cases where the input belongs to the labeled set or to the unlabeled one. The losses calculated were summed using the loss weight defined by Equation 4.1. We tested the methods using softmax and crossentropy loss functions in order to have a more complete comparison.

We performed two kind of experiments repeated three times with random labeled images for training purpose. In the first experiment (see results in Table 4.2) labeled images were balanced per camera and per class. We decided to balance labeled images per camera to give the possibility learn in equal way all the geometrical variabilities of parking spaces. In the second experiment (see results in Table 4.3) we balanced the labeled images only per class to verify the robustness of the classifier.

The networks were trained for 30 epochs calculated over the training set data. Pseudo-label approach takes more iterations than fine-tuning. The learning rate was set at 0.0005. As we can see in Table 4.2 and Table 4.3 results of pseudo-label and fine-tuned AlexNet are comparable over PKLot dataset. Fine-tuning works even better at lower training samples. On the other hand, our first experiment on the PSD dataset obtained results with a huge margin in favour of the fine-tuned AlexNet (see Table 4.2 and Table 4.3). The standard deviation among different runs is really high. This suggested us that the result of Pseudo-label is highly depending on the input. The main difference between the experiments with the two dataset is related on the random choice of data to be labeled. In fact, PKLot unlabeled subset was balanced in terms of classes (i.e. 36000 Empty spaces, 36000 Non empty spaces), whereas the PSD unlabeled subset was not balanced (122903 empty vs 21097 Non-empty spaces). To prove our intuition we then took a subset of PSD dataset balanced per class. In this case the dataset was composed by 42000 images (i.e. 21000 Empty spaces, 21000 Non empty spaces) and corresponding training data considered for the training were

Table 4.2: Results with training balanced per camera and class

Dataset	**Method**	**Loss**	**0.17%**	**1%**	**1.7%**	**5%**
PKLot	finetuning	crossentropy	97.35% ± 2.17	99.40% ± 0.04	99.54% ± 0.04	99.76% ± 0.02
	pseudolabel	crossentropy	94.85% ± 1.81	98.90% ± 0.13	99.35% ± 0.17	99.77% ± 0.04
	finetuning	softmax	97.35% ± 2.17	99.40% ± 0.04	99.54% ± 0.04	99.76% ± 0.02
	pseudolabel	softmax	97.03% ± 0.79	99.07% ± 0.17	99.32% ± 0.37	99.81% ± 0.06
PSD	finetuning	crossentropy	99.02% ± 0.14	99.46% ± 0.15	99.52% ± 0.01	99.73% ± 0.01
	pseudolabel	crossentropy	95.76% ± 1.60	99.25% ± 0.04	99.38% ± 0.13	99.81% ± 0.02
	finetuning	softmax	99.02% ± 0.14	99.46% ± 0.15	99.52% ± 0.01	99.73% ± 0.01
	pseudolabel	softmax	96.89% ± 0.94	99.34% ± 0.07	99.35% ± 0.13	99.81% ± 0.04
PSD*	pseudolabel	crossentropy	98.24% ± 0.13	99.06% ± 0.02	97.24% ± 0.56	97.86% ± 0.02
	pseudolabel	softmax	97.55% ± 0.56	98.82% ± 0.11	97.45% ± 0.24	97.93% ± 0.22

Table 4.3: Results with training balanced per class

Dataset	**Method**	**Loss**	**0.17%**	**1%**	**1.7%**	**5%**
PKLot	finetuning	crossentropy	96.46% ± 0.49	98.36% ± 0.33	98.70% ± 0.01	99.02% ± 0.04
	pseudolabel	crossentropy	15.24% ± 0.67	17.13% ± 0.91	20.65% ± 6.00	14.65% ± 0.00
	finetuning	softmax	96.39% ± 0.26	98.25% ± 0.33	98.47% ± 0.08	99.00% ± 0.15
	pseudolabel	softmax	15.24% ± 0.67	17.13% ± 0.91	20.65% ± 6.00	14.65% ± 0.00
PSD	finetuning	crossentropy	96.92% ± 0.13	98.24% ± 0.05	98.59% ± 0.08	99.05% ± 0.06
	pseudolabel	crossentropy	15.14% ± 0.55	51.69% ± 26.55	61.78% ± 33.33	38.22% ± 33.33
	finetuning	softmax	96.83% ± 0.55	98.10% ± 0.39	98.68% ± 0.17	99.12% ± 0.11
	pseudolabel	softmax	15.14% ± 0.55	51.69% ± 26.55	61.78% ± 33.33	38.22% ± 33.33
PSD*	pseudolabel	crossentropy	98.50% ± 0.12	98.99% ± 0.05	97.80% ± 0.21	98.19% ± 0.28
	pseudolabel	softmax	98.23% ± 0.29	98.99% ± 0.17	97.89% ± 0.30	98.01% ± 0.47

computed in percentage as before (see PSD* in Table 4.1). Results over PSD* in Table 4.2 and Table 4.3 show that the behavior of pseudo-label changed obtaining results similar to the fine-tuned model. In sum, the experiments pointed out that the supervised method (AlexNet plus fine-tuning) outperforms the semi-supervised one (Pseudo-label) in all cases, obtaining very high accuracy (over 96% with few images as training). Moreover good results can be obtained with Pseudo-label only when the dataset to be classified is balanced in terms of samples per classes, which is a prior knowledge too difficult have in real applications.

4.4 Discussions

In this chapter we have compared supervised vs semi-supervised approaches on the problem of parking lots classification. Results shown that the supervised approach AlexNet with fine-tuning outperforms pseudo-label method. Moreover the pseudo-label suffers when the dataset to be classified is composed by samples unbalanced

with respect to the classes. In the next chapter we study the parking management problem as a counting problem.

Chapter 5

Counting Cars And Non-empty Parking Space In a Monitored Area

In this chapter, we investigate the use of vision-based techniques to estimate the occupancy status of parking areas. In particular, we consider two scenarios:

1. *Stall-Based Occupancy Estimation*: in this scenario, we assume parking stalls to be marked on the ground. This allows to formulate assumptions on where cars are expected to be parked;

2. *Stall-Free Occupancy Estimation*: in this more challenging scenario, no assumption on the presence or location of parking stalls is made.

To fairly evaluate approaches tailored to the two scenarios in a unified way, we address the considered task as a counting problem, i.e., the occupancy status of the parking area is determined counting the number of non-empty parking spaces (in the stall-based scenario) or the number of cars (in the stall-free scenario). If the capacity of the parking area is known, as in the most of real cases, is then trivial to determine the number of available free parking spaces. It should be noted that, by considering and benchmarking the problem as a counting task (rather than for instance as a stall classification task), we are able to gain independence from the specific approach employed to address the problem. Taking advantage of this formulation, we benchmark different vision-based approaches such as image classification, object detection and semantic segmentation to estimate the occupancy status of parking areas.

Our investigation is related to previous literature addressing vision-based counting in specific application contexts. For instance, some methods tackled the problem of counting people in crowded scenes [53, 54, 55, 56, 57], cells in biological images [56], bacterial colonies [58], penguins [59], etc.

According to [60], counting methods can be generally divided into three groups:

- *counting by detection*: these methods use object detection to count extensively [54];
- *counting by clustering*: these methods assume the presence of individual entities presenting unique yet coherent patterns which can be clustered to approximate the final number of instances [61];
- *counting by regression*: these methods count entities by learning a direct mapping from low-level imagery to numbers [53, 56, 62, 63];

5.1 Methods

We benchmark different approaches to estimate the occupancy status of parking areas from images. Specifically, we consider solutions based on three popular computer vision technologies: image classification, object detection and semantic segmentation. As discussed in the previous sections, we will consider two main scenarios: stall-based occupancy estimation and stall-free occupancy estimation.

5.1.1 Stall-Based Occupancy Estimation

The approaches considered in this scenario assume that the position of each stall is known in advance (see Figure 5.1). It should be noted that, since the camera is fixed, labeling the position of the stalls is part of a calibration process which needs to be performed only once, when the camera is installed. Since the position of each stall is known, it is natural to find the number of available parking spaces by determining the status of each stall (free/occupied). As anticipated, we explore three possible approaches based on image classification, object detection and semantic segmentation for this scenario.

Figure 5.1: An example image acquired in a real parking lot using a fixed camera. Parking stalls are highlighted in green (the image is best seen in digital format). To count non-empty parking spaces, we assume, as in a real scenario, that the positions of the stalls are known in advance.

The approach based on image classification works as follows. Given an input image, we sample around each stall the smallest rectangular image patch containing the entire stall. Each extracted patch is labeled as "empty" or "full" depending on the occupancy status of the related stall. A classifier is hence trained to discriminate between "empty" and "full" stalls. At inference time, the trained classifier is used to determine the status of each stall in order to obtain the number of non-empty parking spaces. This approach is illustrated in Figure 5.2 (a).

The approach based on object detection works as follows. An object detector is employed to localize all vehicles present in the input image. At inference time, all bounding boxes detected with a score lower than a given threshold d_1 are discarded. The Intersection Over Union (IoU) measure between each stall and each retained bounding box is then computed. A stall is deemed to be occupied if the IoU between the stall area and at least one of the detected bounding boxes is higher than a given threshold d_2. The thresholds d_1 and d_2 are optimized empirically on a validation set as detailed in Section 5.3. The approach is illustrated in Figure 5.2 (b).

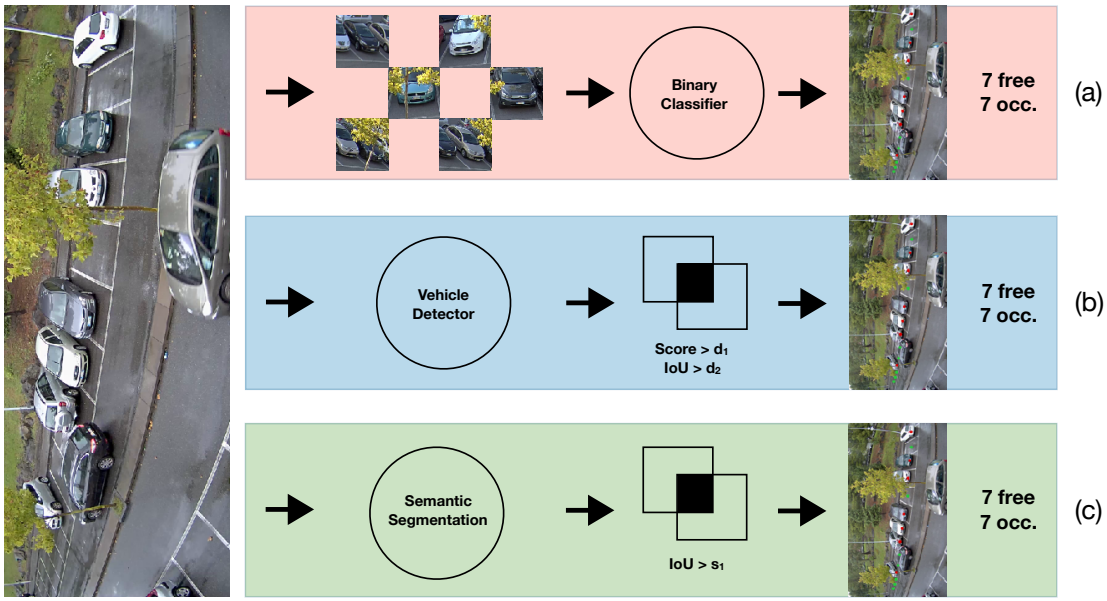


Figure 5.2: Considered methods for stall-based occupancy status estimation. The figure illustrates methods based on (a) image classification, (b) object detection, (c) semantic segmentation. See text for details. The image is best seen in digital version.

The approach based on Semantic Segmentation works as described in the following. We first train a Semantic Segmentation algorithm to discriminate between two classes: vehicle and background. At inference time, a stall is classified as "occupied" if the pixel-based IoU between the stall and the segmentation mask is higher than a given threshold s_1. The thresholds s_1 is optimized empirically on a validation set as detailed in Section 5.3. This method is illustrated in Figure 5.2 (c).

5.1.2 Stall-Free Occupancy Estimation

The approaches focusing on stall-free occupancy estimation do not make any assumption on the existence or position of stalls in the scene. Since the image classification approach cannot be used in this scenario, we consider two different approaches based on object detection and image segmentation. We assume that each input image is provided with a Region of Interest (RoI) indicating where the cars should be counted. This allows to exclude areas of the image where cars may transit, such as the road. Figure 5.3 reports an example of a RoI defined on an image.

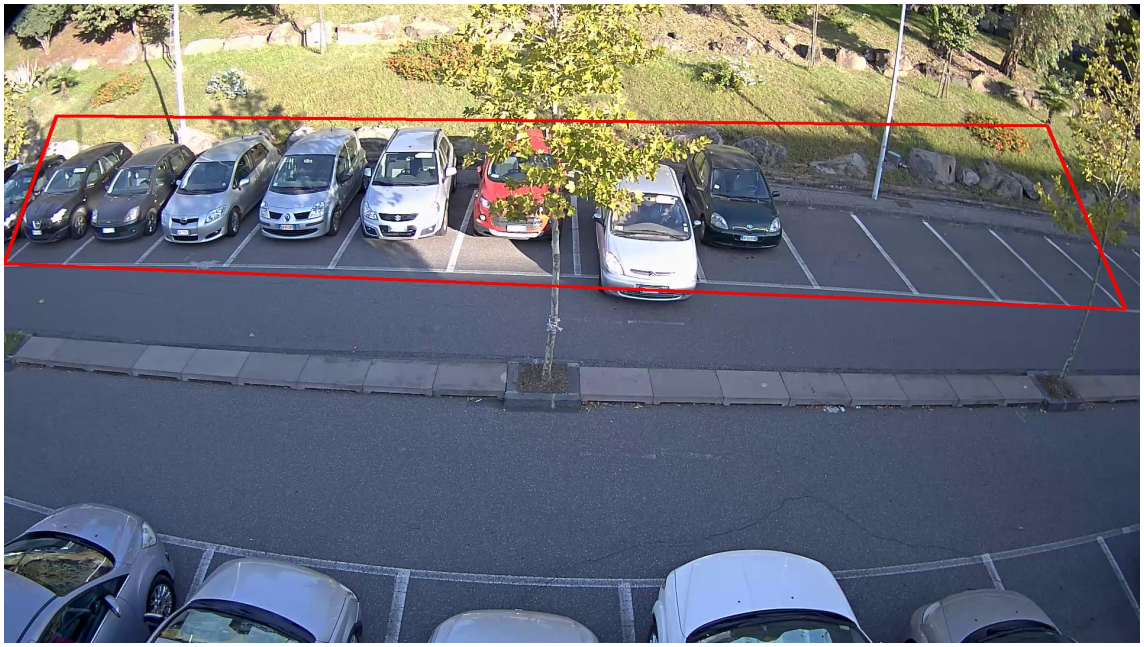


Figure 5.3: An example of Region of Interest RoI defined on an image.

The stall-free occupancy estimation method based on object detection adopts the "counting by detection". The method works as follows. An object detector is used to localize all cars present in the scene. Bounding boxes with detection score lower than the threshold r_1 or with IoU score with the RoI lower than a threshold r_2 are discarded. The thresholds r_1 and r_2 are optimized empirically on a validation set as detailed in Section 5.3. We obtain the total number of cars present in the RoI by counting all retained bounding boxes. Figure 5.4 (a) illustrates this method.

The method based on semantic segmentation for stall-free occupancy estimation works as follows. An image segmentation algorithm is used to segment cars versus background. The segmentation mask is hence fed to a multi-class classifier, which is trained to predict a class label comprised between 1 and n, where n is the capacity of the parking area (i.e., the maximum number of cars which can be parked in the area). The predicted class corresponds to the number of cars present in the RoI. It should be noted that this composite architecture can be used to predict the number of cars directly from the input image frame. Figure 5.4 (b) illustrates this approach.

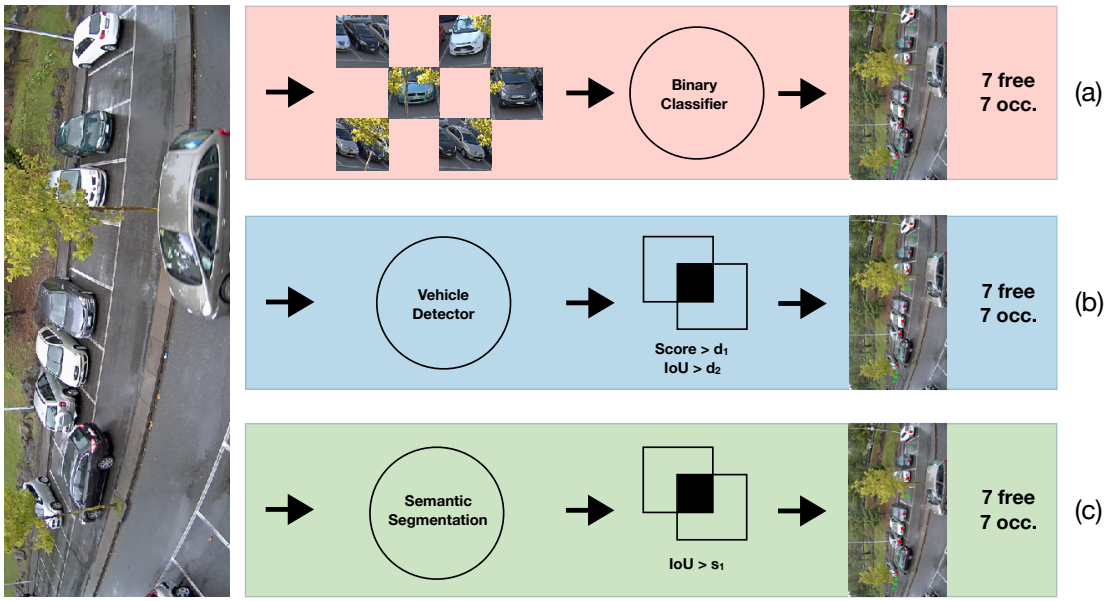


Figure 5.4: Considered methods for stall-free occupancy status estimation. The figure illustrates methods based on (a) object detection, (b) semantic segmentation. See text for details. The image is best seen in digital version.

Temporal Smoothing

When counting from videos, it is possible to take advantage of temporal consistence to reduce random flicker in the predictions. In particular, the number of cars in a parking lot is expected to change smoothly between consecutive observations. We investigate a simple approach to enforce a temporal constraint, which consists in averaging the predicted count using a sliding window.

5.2 Dataset

We perform our experimental analysis on a dataset of 11,066 images captured in our living lab which is located at the campus of the University of Catania (Figure 5.5). For each image of the dataset, we manually collected ground truth labels in the form of semantic segmentation masks, bounding boxes and parking stall configurations.

The dataset has been acquired using three Full-HD cameras looking at different parking spaces. The three cameras are referred to as "Camera 1", "Camera 2" and "Camera 3". "Camera 1" observes 12 parking spaces (Figure 5.6), "Camera 2"

Figure 5.5: A satellite image [64] of the monitored parking area located at the campus of the University of Catania.

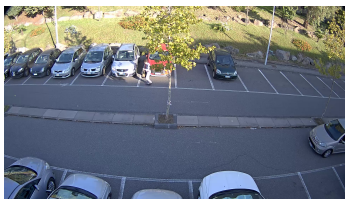


Figure 5.6: Camera 1 observes 12 parking spaces.

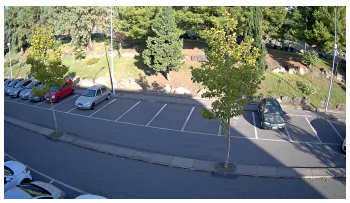


Figure 5.7: Camera 2 observes 14 parking spaces.

Figure 5.8: Camera 3 observes 12 parking spaces.

monitors 14 parking spaces (Figure 5.7), and "Camera 3" acquires images of 12 parking spaces (Figure 5.8). Given the different viewpoints of the cameras, the acquired scenes are characterized by different scene geometries. We recorded two long videos per camera at $1 fps$. The two videos have been acquired in different days and care has been taken to cover many configurations of the parking spaces,

Table 5.1: Videos contained in the dataset, along with the corresponding number of labeled frames.

Camera	**Video**	**Number of Frames**
Camera 1	Video 1.1	1,801
	Video 1.2	1,801
Camera 2	Video 2.1	1,801
	Video 2.2	2,241
Camera 3	Video 3.1	1,321
	Video 3.2	2,101
	Total	11,066

including the cases in which the observed parking area was empty and fully occupied. Table 5.1 summarizes the videos contained in the dataset and reports the number of frames of the considered videos.

Each frame of the dataset has been manually labeled to report:

- a Region of Interest to identify the area within which the parking spaces are comprised;
- the total number of cars present in the monitored parking space;
- a bounding box around each car inside the monitored parking space (see Figure 5.9);
- a weak semantic segmentation where the pixel-level masks are created from bounding boxes. All the pixels inside a bounding box are considered as belonging to the "car" class, whereas every pixel outside the bounding box is considered as background (see Figure 5.10 for an example);
- a binary vector, each component of which represents the status of the i-th parking space as empty (0) or non-empty (1);
- the coordinates of the four corners for each stall present in the frame (see Figure 5.1).

We propose two different ways of splitting the data into training and testing sets. The first split assumes that training and testing data have been acquired using a single camera. This gives rise to 6 different data subsets (one for each camera),

Figure 5.9: Bounding boxes annotated (in green) in each frame of the dataset.

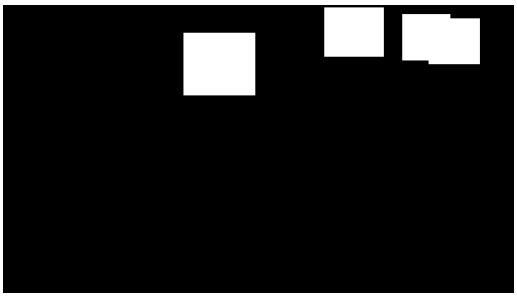


Figure 5.10: An example of weak semantic segmentation label obtained from the annotated bounding boxes.

Table 5.2: Data subsets arising from the two considered data splits.

Subset	**Training Data**	**Testing Data**
Subset 1a	Video 1.1	Video 1.2
Subset 1b	Video 1.2	Video 1.1
Subset 2a	Video 2.1	Video 2.2
Subset 2b	Video 2.2	Video 2.1
Subset 3a	Video 3.1	Video 3.2
Subset 3b	Video 3.2	Video 3.1
Subset A	Videos 1.1, 2.1, 3.1	Videos 1.2, 2.2, 3.2
Subset B	Videos 1.2, 2.2, 3.2	Videos 1.1, 2.1, 3.1

where one of the two videos is used for training, and the other one is used for testing (subset "Nx" in Table 5.2). These 6 subsets are intended to assess the performance of methods when exposed to data acquired from a single camera.
The second data split assumes that both training and test data have been acquired using the three cameras. In this case, we obtain two subsets (subset "X" in Table 5.2), where data acquired using the three cameras, but belonging to one of the two videos is used for training, while the remaining is used for test.
The above two subsets are intended to assess the ability of methods to generalize to different scenes.

In order to pre-train the network to perform semantic segmentation and counting cars in a given RoI, we built a synthetic dataset of $100,000$ plausible semantic segmentation masks. Each image has been created generating a random number of bounding boxes placed according to the geometry of the stalls coming from the three cameras. To include more variability, we jitter both position and size of the

bounding boxes. Specifically, the size of each bounding box is randomly selected to match 70%-110% of the original size and the bounding box is randomly positioned within $+15/-15$ pixels with respect to the original position. We used 70% of the frames for training and the remaining 30% for test.

5.3 Experimental Settings

We implement the image classification component fine-tuning a VGG16 Convolutional Nerual Network (CNN) [23] pre-trained on ImageNet [65] to discriminate between "empty" and "full" stalls. The extracted image patches from the frames constitute in total $17,688$ samples for *Video 1.1*, $17,712$ samples for *Video 1.2*, $20,636$ samples for *Video 2.1*, $25,676$ samples for *Video 2.2*, $13,032$ samples for *Video 3.1*, $20,556$ samples for *Video 3.2*. For every subset, a different model has been trained and the fine-tuning process has been carried out for 10 epochs. Test accuracies for the different data subsets are reported in the first column of Table 5.3. Given the dependence on scene geometry (image patches are cropped according to the geometry of the scene), the classifier achieves best results when a single geometry is considered (first six rows of Table 5.3), while performance decreases when different geometries are mixed (last two rows of Table 5.3).

The car detector is a fine-tuned FasterRCNN [26] object detector based on VGG16. The detector has been fine-tuned starting from the weights of VGG16 trained on ImageNet. The training process has been carried out for $70,000$ iterations using a batch size of 1. As for the classification, we trained a separate model for each data subset. The second column of Table 5.3 reports mAP values on the test sets for each of the considered data subsets. As can be noted, the detector benefits from the larger training sets included in Subset A and Subset B.

The Semantic Segmentation method is implemented fine-tuning SegNet [66] on the considered dataset. In particular, we fine-tuned the network starting from weights pre-trained on ImageNet to segment cars vs background. To gather training data, we considered the weak semantic labels included in the collected dataset, as it is described in Section 5.2. The pixel accuracy of each model is reported in the third column of Table 5.3. Also in this case, the results highlight how the method benefits from the larger training sets of Subset A and Subset B (last two rows of Table 5.3).

Table 5.3: Performance of the three trained components on the test sets. Performance of the binary stall classifier is measured using accuracy (fraction of correctly classified stalls). Performance of the car detector is measured using standard mean Average Precision (mAP). Performance of semantic segmentation is measured using pixel accuracy (fraction of correctly classified pixels).

Subset	VGG16 Stall Classifier (Accuracy)	FasterRCNN Car Detector (mAP)	SegNet Semantic Segmentation (Pixel Accuracy)
Subset 1a	0.991	0.224	0.892
Subset 1b	0.987	0.381	0.857
Subset 2a	0.986	0.358	0.595
Subset 2b	0.988	0.185	0.766
Subset 5a	0.949	0.228	0.715
Subset 5b	0.989	0.340	0.902
Subset A	0.911	0.551	0.896
Subset B	0.952	0.698	0.892

To train the composite architecture which combines semantic segmentation and classification to count cars, we used the following procedure. We first trained a classification architecture based on LeNet [19] to classify the $100,000$ synthetic images contained in the proposed dataset. Here, we set the number of classes predicted by the final classifier to 14, which is the maximum number of parking spaces seen by the cameras. Specifically, the network has been trained for $50,000$ iterations using a batch size of 20 images. We hence concatenated the LeNet classification architecture to the previously trained SegNet architecture and fine-tuned the composite architecture for $20,000$ iterations with a batch size of 1 image. The fine-tuning procedure has been carried out independently for each data subset.

5.3.1 Optimization of Thresholds and Seletion of RoI

The methods discussed in the previous sections make use of different thresholds to classify parking stalls as "empty" or "occupied" and to count cars. We set such thresholds to the values which optimize the performance of the considered methods on a validation set which is formed randomly selecting 15% of the training samples. The search for optimal values is performed independently on each data subset.

Specifically, when counting non-empty parking spaces using the pipeline based on object detection, we chose the values of d_1 and d_2 which maximize stall classification

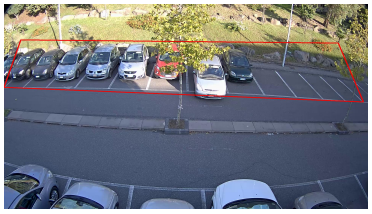


Figure 5.11: Region of Interest (RoI) for Camera 1.

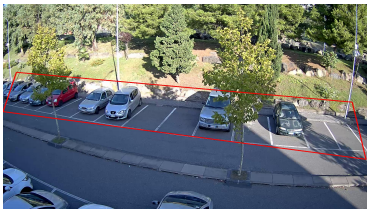


Figure 5.12: Region of Interest (RoI) for Camera 2.

Figure 5.13: Region of Interest (RoI) for Camera 3.

accuracy on the validation set (see Figure 5.14). Similarly, when counting non-empty parking spaces using semantic segmentation, we chose the value of the threshold s_1 which maximizes stall classification accuracy on the validation set (see Figure 5.15). When counting cars using the pipeline based on image segmentation, we chose the values of r_1 and r_2 which minimize the Mean Absolute Error (MAE) on the validation set. Figures 5.11, 5.12 and 5.13 show the RoIs considered in our experiments.

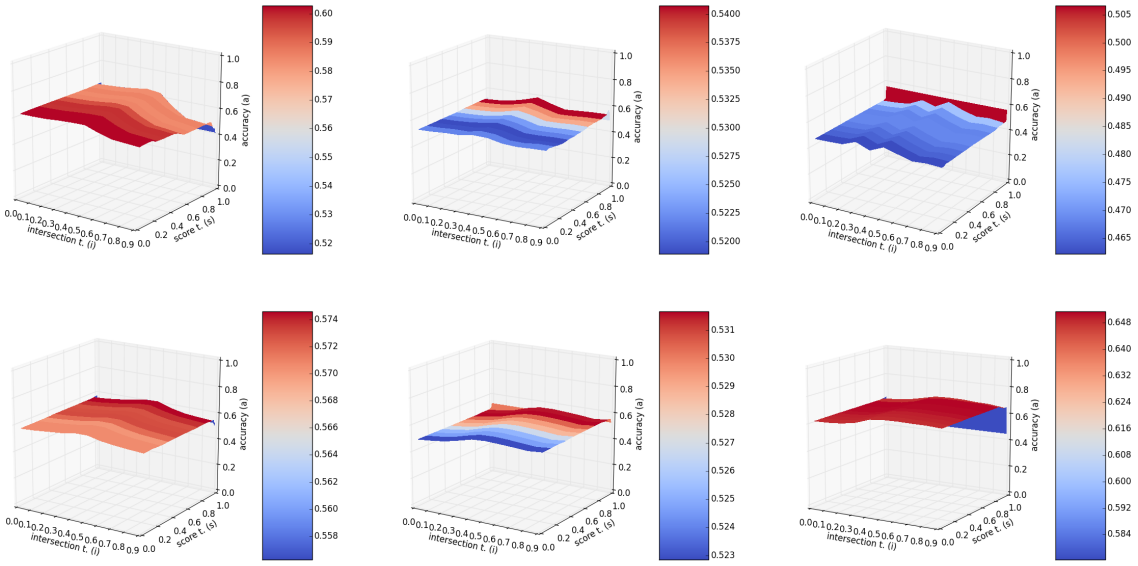


Figure 5.14: Example of hyperparameters search over validation test for counting empty and non-empty spaces using car detection on single camera subset. Each graph shows the accuracy over validation set varying d_1 and d_2 thresholds

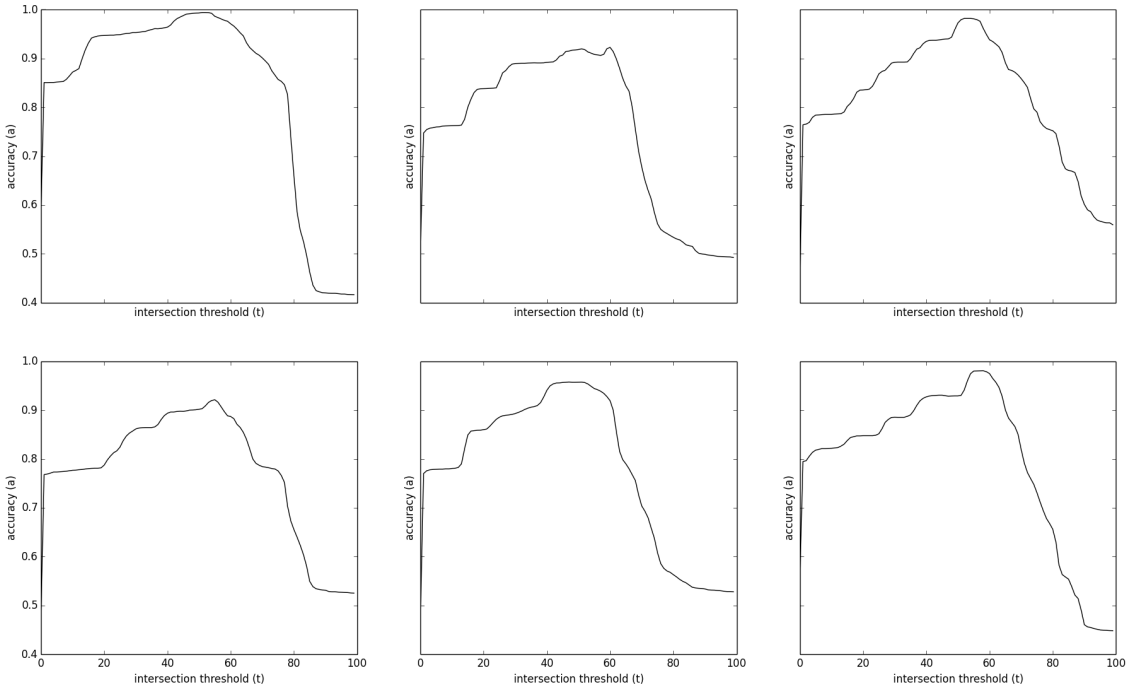


Figure 5.15: Example of hyperparameters search over validation test for counting empty and non-empty spaces using semantic segmentation on single camera subset. Each graph shows the accuracy over validation set varying the s_1 threshold

5.3.2 Evaluation Measures

To assess the discrepancy between predicted counts and ground truth counts, we evaluate the investigated approaches by computing the Absolute Errors (AE). Given a test frame I_i, the predicted count $\hat{y}_i$, and the ground truth count y_i, we compute the absolute error corresponding to I_i as follows:

$$AE_i = |y_i - \hat{y}_i| \tag{5.1}$$

To evaluate the performance on a set of test frames $\mathcal{I} = \{I_i\}_{i=1}^{N}$, we also compute statistics of the AE values computed for each frame, including minimum, maximum, median and mean. In particular, among the other measures, we consider the classic Mean Absolute Error (MAE):

$$MAE(\mathcal{I}) = \frac{1}{N}\sum_{i}^{N} AE_i = \frac{1}{N}\sum_{i}^{N} |\hat{y}_i - y_i| \tag{5.2}$$

It should be noted that the absolute errors and the derived statistics are easy to interpret, as they are expressed with the same unit measure of the original data, e.g. a method reporting a MAE equal to 1 is, in average, overestimating or underestimating the count by 1 unit.

Apart the MAE measure, the performances of the different components employed in the investigated methods (i.e., image classification, object detection and image segmentation components) are evaluated by the most appropriate measures. Specifically, we evaluate image classification using accuracy (fraction of correctly classified images), object detection using mean Average Precision (mAP) as proposed by [67], and image segmentation using pixel accuracy (fraction of correctly classified pixels), as shown in the results reported in Table 5.3.

5.4 Results

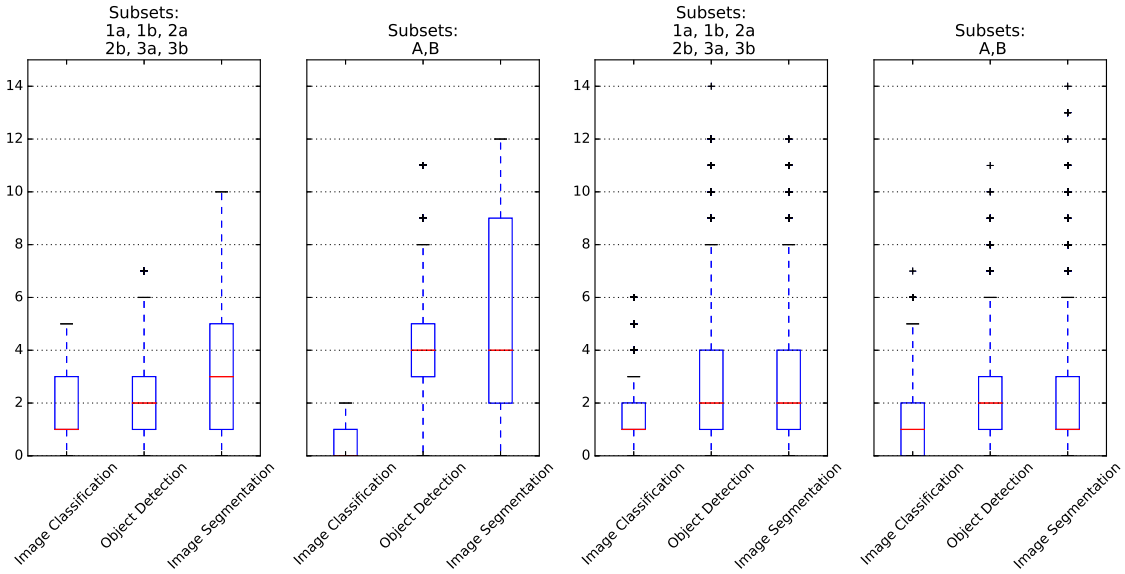


Figure 5.16: Box plots of tests for counting non-empty spaces

Figure 5.17: Box plots of tests for counting cars.

The third column of Table 5.4 reports Mean Absolute Error, Maximum Absolute Error and Median Median Absolute Error for the problem of counting in the Stall-Based scenario.

Table 5.4: Mean Absolute Errors for both counting tasks. Median and Maximum Absolute Errors in parentheses. Minimum is always 0.00. Best results for each data subset are reported in bold numbers.

Subset	Method	Stall-Based	Stall-Free
Subsets: 1a, 1b, 2a 2b, 3a, 3b	Image Classification	**1.63 (1.00/5.00)**	*1.57 (1.00/6.00)*
	Object Detection	2.15 (2.00/7.00)	2.83 (2.00/14.00)
	Semantic Segmentation	3.32 (3.00/10.00)	**2.58 (2.00/12.00)**
Subset 1a	Image Classification	**1.48 (1.00/3.00)**	*1.44 (1.00/6.00)*
	Object Detection	4.20 (4.00/7.00)	**2.38 (2.00/11.00)**
	Semantic Segmentation	3.72 (5.00/7.00)	2.84 (3.00/9.00)
Subset 1b	Image Classification	**0.54 (0.00/2.00)**	*0.56 (0.00/4.00)*
	Object Detection	2.26 (3.00/5.00)	3.43 (3.00/11.00)
	Semantic Segmentation	4.79 (4.00/10.00)	**3.39 (3.00/12.00)**
Subset 2a	Image Classification	2.49 (3.00/3.00)	*2.37 (2.00/6.00)*
	Object Detection	**1.54 (1.00/5.00)**	2.30 (2.00/10.00)
	Semantic Segmentation	2.02 (2.00/5.00)	**1.50 (1.00/6.00)**
Subset 2b	Image Classification	3.22 (4.00/5.00)	*3.15 (4.00/6.00)*
	Object Detection	**2.04 (2.00/4.00)**	5.03 (5.00/14.00)
	Semantic Segmentation	5.53 (7.00/9.00)	**4.84 (5.00/11.00)**
Subset 3a	Image Classification	**0.88 (1.00/2.00)**	*0.85 (1.00/5.00)*
	Object Detection	1.28 (1.00/4.00)	**1.92 (2.00/7.00)**
	Semantic Segmentation	2.76 (1.00/6.00)	**1.92 (2.00/7.00)**
Subset 3b	Image Classification	**0.86 (1.00/3.00)**	*0.74 (1.00/2.00)*
	Object Detection	1.82 (2.00/3.00)	1.98 (1.00/9.00)
	Semantic Segmentation	0.88 (1.00/3.00)	**0.95 (1.00/3.00)**
Subsets: A, B	Image Classification	**0.56 (0.00/2.00)**	*1.16 (1.00/7.00)*
	Object Detection	4.23 (4.00/11.00)	**1.95 (2.00/11.00)**
	Semantic Segmentation	5.38 (4.00/12.00)	1.97 (1.00/14.00)
Subset A	Image Classification	**0.40 (0.00/1.00)**	*0.87 (1.00/5.00)*
	Object Detection	4.09 (4.00/8.00)	**1.77 (1.00/9.00)**
	Semantic Segmentation	5.49 (5.00/12.00)	1.79 (1.00/9.00)
Subset B	Image Classification	**0.68 (0.00/2.00)**	*1.39 (1.00/7.00)*
	Object Detection	4.34 (4.00/11.00)	**2.10 (2.00/11.00)**
	Semantic Segmentation	5.29 (4.00/12.00)	2.11 (1.00/14.00)

The table reports the results for the different data subsets, as well ass for their aggregation. In Figure 5.16, we report the box plots for the Absolute Errors obtained for the aggregated data subsets. The best performances are achieved by the method based on Image Classification. This method always obtains a minimum Absolute Errors equal to 0 and a maximum Absolute Errors values not exceeding 5 units. Median errors are often close to zero. The Mean Absolute Errors of Image Classification methods are overall significantly lower than the others both in the

case of single camera tests (Subsets 1a to 3b) and multiple camera tests (Subsets A and B). This observation is made particularly clear by Figure 5.16, in which the box plots related to the Image Classification method exhibit median values and quartile positions lower than the others. The method based on Object Detection is second best in all camera scenarios and it is followed by the method based on Semantic Segmentation. Interestingly, the method based on Image Classification benefits from the presence of different geometries in the training set, allowing to further lower the MAE of 1.63 to 0.68. On the contrary, the other methods based on object classification and Semantic Segmentation achieve much worse results (from 2.15 to 4.23 in the case of Object Detection and from 3.32 to 5.38 in the case of Semantic Segmentation). This observation suggests that the method based on Image Classification is more capable to generalize, while the other two methods probably suffer from overfitting.

The fourth column of Table 5.4 and Figure 5.17 report the statistics of Absolute Error values and box plots related to the stall-free scenario. Along with the results of the methods based on object detection and semantic segmentation, the results of the "Image Classification" method are also reported for reference. In this latter case, the number of cars in the scene is estimated classifying each stall as whether "free" or "occupied". It should be noted that the performance of the "Image Classification" method is reported for reference only and it is not directly comparable to those of the other methods in this scenario. As one could expect, the method based on Image Classification achieves the best results (MAE equal to 1.57 in the case of a single camera geometry and 1.16 for multiple camera geometres). This suggest that, when stalls are present and their location is known (i.e., in most of the real cases), this information can be exploited to count cars. Among the two competing stall-free methods, the one based on Semantic Segmentation outperforms Object Detection when multiple geometries are considered. Interestingly, all the methods benefit from the presence of different camera geometries in this experiment.

As a general remark, when the location of parking stalls is not known and the method based on Image Classification cannot be applied, a method based on Semantic Segmentation is probably to be preferred given the comparable MAE in both scenarios (2.58 for Semantic Segmentation vs 2.83 for image detection when a single

camera geometry is considered and 2.11 vs 2.10 for multiple geometries) and the lower median absolute error (1 vs 2 for multiple geometries).

We finally investigate the application of temporal smoothing to improve the results of methods for car counting. We consider the stall-free scenario, since it is the most general and challenging one. Figure 5.18 shows the Mean Absolute Error values obtained performing temporal smoothing with different window sizes on the single camera subsets, whereas Figure 5.19 reports the same results averaged over all data subsets. Since all videos are acquired at a frame rate of 1 fps, window sizes are measured in seconds. It should be noted that temporal smoothing can be applied only to videos, therefore we exclude from this analysis Subsets A and B which contain several videos.

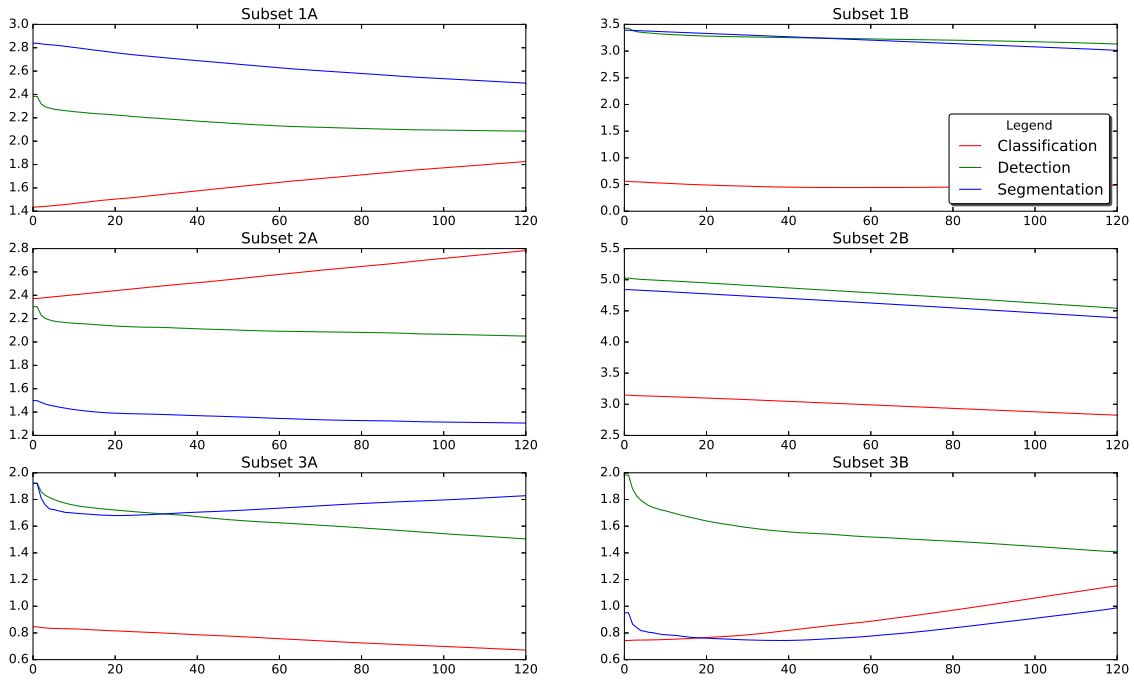


Figure 5.18: Temporal smoothing: for each subset we show the MAE with different averaging frames. The x axes represent the size of the temporal window in seconds, while the y axes represent the obtained MAE.

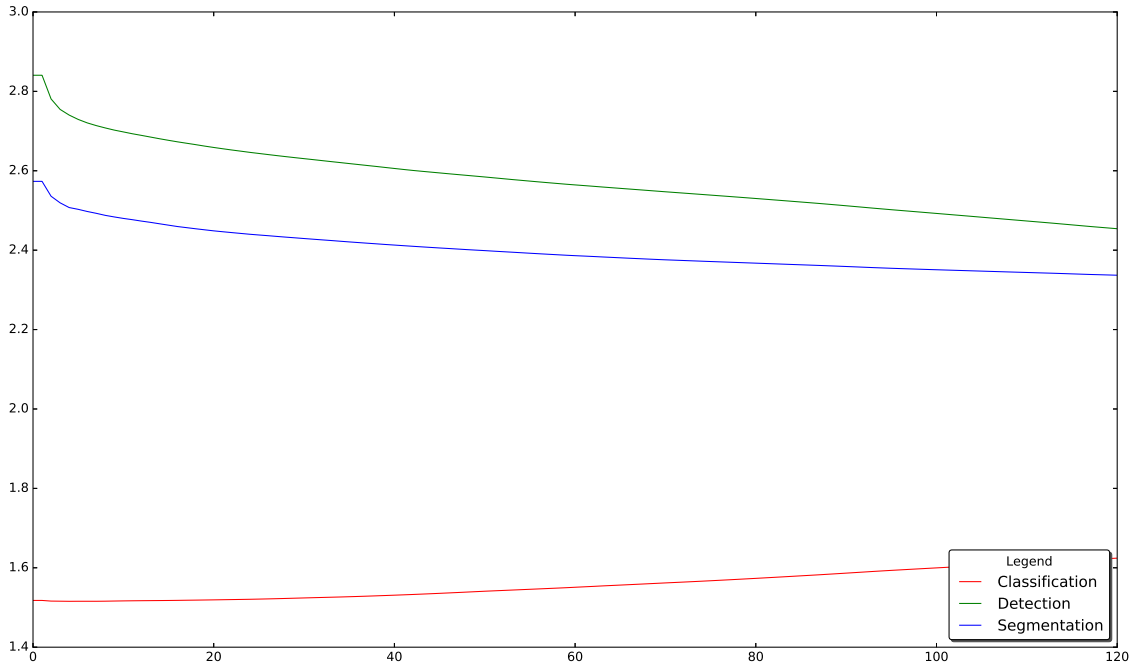


Figure 5.19: Temporal smoothing results averaged over the different data subsets. The x axes represent the size of the temporal window in seconds, while the y axes represent the obtained average MAE.

The results reported in Figure 5.19 show that both methods based on Object Detection and Image Classification benefit in average from temporal smoothing. This observation is true for all data subsets in the case of Object Detection, while it is not true for Subsets 3a and 3b in the case of Semantic Segmentation (see Figure 5.18). In general, the method based on Image Classification does not benefit from temporal smoothing (see Figure 5.19 and Subset 1a, 1b, 2a, 2b, 3b in Figure 5.18). Figure 5.20 shows a visual example of counting in the stall-based scenario. A complete video to better assess the results is available at http://iplab.dmi.unict.it/ParkSmartCounting.

5.5 Discussions

This chapter has investigated and compared different methods to address the problem of estimating the occupancy status of parking areas both in a stall-based and stall-free scenarios.

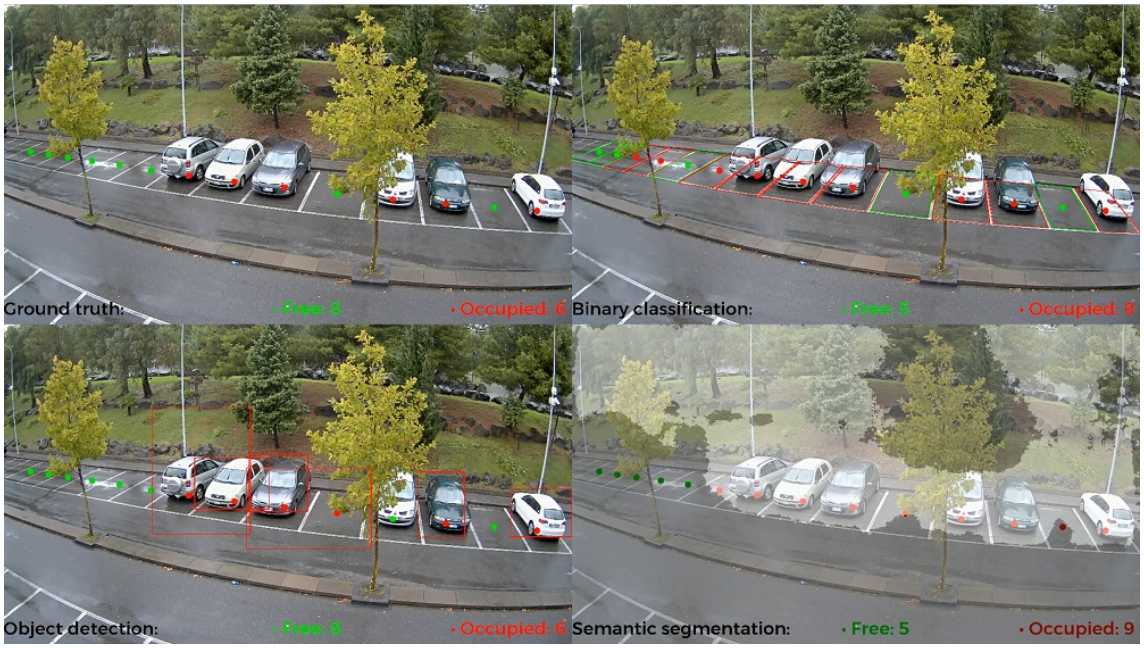


Figure 5.20: An example of discriminating empty and non-empty stalls with considered methods.

Results on a real dataset have shown that, when the geometry of the scene is known, the system can take advantage of classification methods to obtain competitive results. When the configuration of a parking lot is not known, the method based on image segmentation is to be preferred over the method based on object detection. Moreover, we have shown that temporal smoothing can be effectively used to improve results based on object detection and image segmentation, while it is not effective for methods based on image classification. In the next chapter will focus on domain adaptation techniques to transfer acquired knowledge to different parking areas.

Chapter 6

Scene Adaptation for Semantic Segmentation using Adversarial Learning

Semantic image segmentation has a key role in many industrial applications including video surveillance and traffic analysis [5, 6]. Despite semantic segmentation approaches based on deep learning have shown good performance on a variety of tasks (e.g., instance object segmentation [68], scene segmentation [30], etc.) they usually need to be adapted to the considered application domain through a fine-tuning process in order to achieve reasonable results. The fine-tuning process requires the collection and labeling of a large amount of domain-dependent data. In addition, whenever the system setup changes (e.g., cameras are moved or new cameras are installed), the fine-tuning procedure, including the collection and labeling of new data, needs to be repeated. Since collecting and labeling data requires a significant effort, the classic fine-tuning approach prevents industrial applications to scale up. As noted in [69], domain adaptation techniques can be employed in order to limit the amount of labeled data to be used to adapt a pre-trained semantic segmentation algorithm.

In this chapter, we consider a scenario in which a fixed camera is installed to monitor a given area (e.g., a crossroad or a parking lot). When a new camera looking at a different scene or looking at the same scene but from a different point of view is added to the camera network, the semantic segmentation algorithms need to be adapted to the new view through a fine-tuning procedure. We investigate the use of Adversarial Learning [37] to limit the amount of labeled data required for such

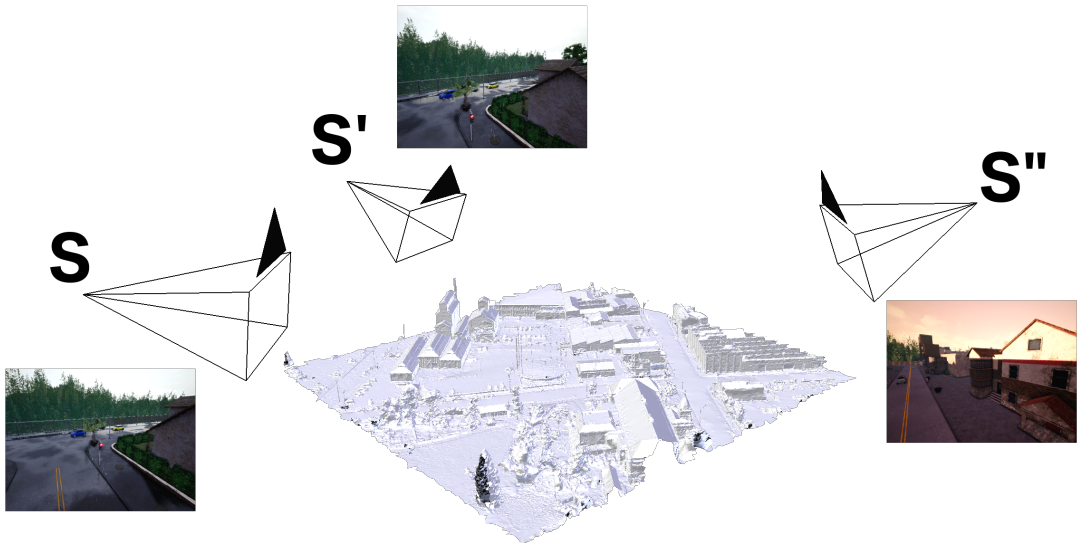


Figure 6.1: Three cameras monitoring different parts of a city. S and S' are related to the same part of the city, but different points of view, whereas S" is an observation of a different part of the city.

adaptation. Specifically, we require that only unlabeled data, effortlessly collected using the new camera, have to be used for the adaptation. The task is framed as a Domain Adaptation problem in which labeled images come from the first camera of the Source domain, and unlabeled data come from the second camera of the Target domain. Figure 6.1[1] illustrates the proposed scene adaptation problem.

To perform the study, we created a dataset of synthetic images using the CARLA Urban Driving Simulator [70]. The dataset consists of images collected from 3 different scene contexts, each acquired from 2 different views. A scene adaptation method based on Adversarial Learning is hence proposed. The method is designed to perform Semantic Segmentation of images of the Target domain after a training phase which exploits labeled images from Source domain and unlabeled images from Target domain. The experiments show that the proposed method outperforms the baselines and achieves results close to (or in some case better than) those obtained by a classic fine-tuning approach.

[1]3D Model from http://s3-ap-southeast-2.amazonaws.com/launceston/atlas/index.html.

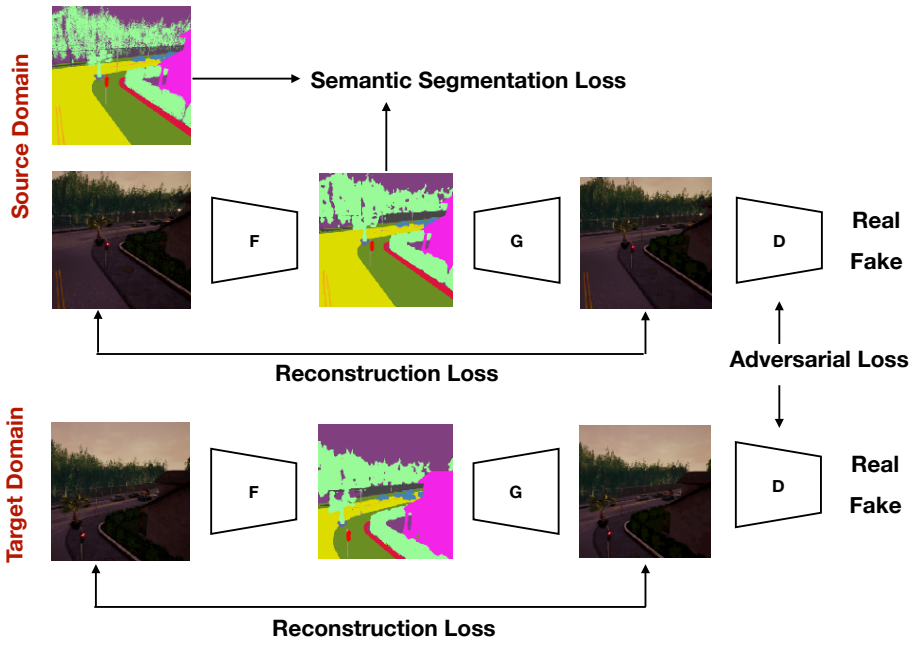


Figure 6.2: The proposed domain adaptation architecture.

6.1 Method

We propose an architecture which allows to train a semantic segmentation network using labeled images from the Source domain and unlabeled images from the Target domain. At test time, the network can be used to predict semantic segmentation masks for new images coming from both domains.

6.1.1 Network Architecture

The proposed architecture is illustrated in Figure 6.2 and consists of three main components: the semantic network to be trained (F), a generative network (G), and a discriminator network (D). The semantic network F is responsible for predicting semantic segmentation masks for images belonging to both Source Domain and Target Domain. The ground truth segmentation masks available for images belonging to the Source Domain are used to provide direct supervision to the semantic network F through a semantic segmentation loss. Since no ground truth segmentation masks are available for images belonging to the Target Domain, supervision

is also provided by enforcing that produced segmentation masks to be suitable for reconstructing the input image. This constraint is implemented using the generative network (G) to reconstruct the input image starting from the predicted segmentation mask and enforcing a reconstruction loss and an adversarial loss making use of the discrimination network (D). This encourages the network to produce meaningful segmentation masks for both domains. The following sections detail the proposed network and loss function.

Semantic Segmentation Network

The semantic segmentation network F predicts a segmentation mask for each input image. We implement this module using the PSPNet architecture proposed Zhao et al. [33] using a pre-trained ResNet backbone [24]. We would like to note that, in principle, any semantic segmentation network trainable end-to-end through gradient descent can be used to implement F in the proposed architecture.

Generative Network

The generative network G reconstructs the input images from the inferred segmentation masks. The network G takes over the pixel-wise class scores produced by the Semantic Network F and generates an RGB image of the same dimensions as the input image. To implement the generative network, we consider the architecture proposed by Johnson et al. [71] as implemented in [43].

Discriminator Network

The discriminator network D is used to train the system using the paradigm of adversarial learning. This is done by training the discriminator to distinguish between real and reconstructed images. The semantic segmentation and generation networks described in the previous section are concurrently trained to "fool" the discriminator as described in [37]. The discriminator network is implemented using a PatchGANs [42, 72, 38] operating on 70×70 overlapping image patches. This patch-level discriminator architecture has fewer parameters than a full-image discriminator and can be applied to arbitrarily-sized images in a fully convolutional fashion [42].

6.1.2 Loss Functions

The overall architecture shown in Figure 6.2 is trained by a composite loss which encourages the network to produce good semantic segmentation masks as well as to obtain good performances in reconstructing the input images.

Semantic Segmentation Loss

The semantic segmentation loss is defined as the cross entropy between the inferred pixel-wise probabilities and the ground truth per-pixel classes. This loss is computed only on images belonging to the Source Domain, since no ground truth segmentation masks are assumed for images of the Target Domain. Specifically, let x be an image from the Source Domain S, let $F(x)$ be the set of scores produced by the semantic segmentation network and let $\mathcal{S}(F(x))$ be the softmax of $F(X)$ computed along class scores independently for each pixel. Let y be the ground truth segmentation mask associated to x, which indicates that a pixel i belongs to class y_i. We define the semantic segmentation loss as follows:

$$\mathcal{L}_{Sem}(F) = \mathbb{E}_{x \sim p_S(x)}[-\sum_i log(\mathcal{S}(F(x))_{i,y_i})] \tag{6.1}$$

where p_S denotes the distribution of the data belonging to the Source Domain, i iterates over the pixels of the training image x, $\mathcal{S}$ denotes the softmax function, and $\mathcal{S}(F(x))_{i,y_i}$ denotes the probability predicted for class y_i at pixel i.

Reconstruction Loss

Inspired by [43], we include a reconstruction loss to encourage the correct reconstruction of the input images, starting from the class scores produced by the semantic segmentation network. The loss is applied to both images belonging to the Source and Target Domains. We use a L_1 loss defined as follows:

$$\mathcal{L}_{Rec}(G, F) = \mathbb{E}_{x \sim p_{ST}(x)}[||G(F(x)) - x||_1] \tag{6.2}$$

where p_{ST} denotes the distribution of data belonging to both the Source and Target Domains, and $|| \cdot ||_1$ denotes the L1 loss.

Adversarial Loss

The adversarial loss is employed to encourage the network G in Figure 6.2 to produce better reconstructions and overcome the common limitations of regression losses [73]. This loss is applied to samples belonging to both Source Domain and Target Domain. The loss is defined as follows:

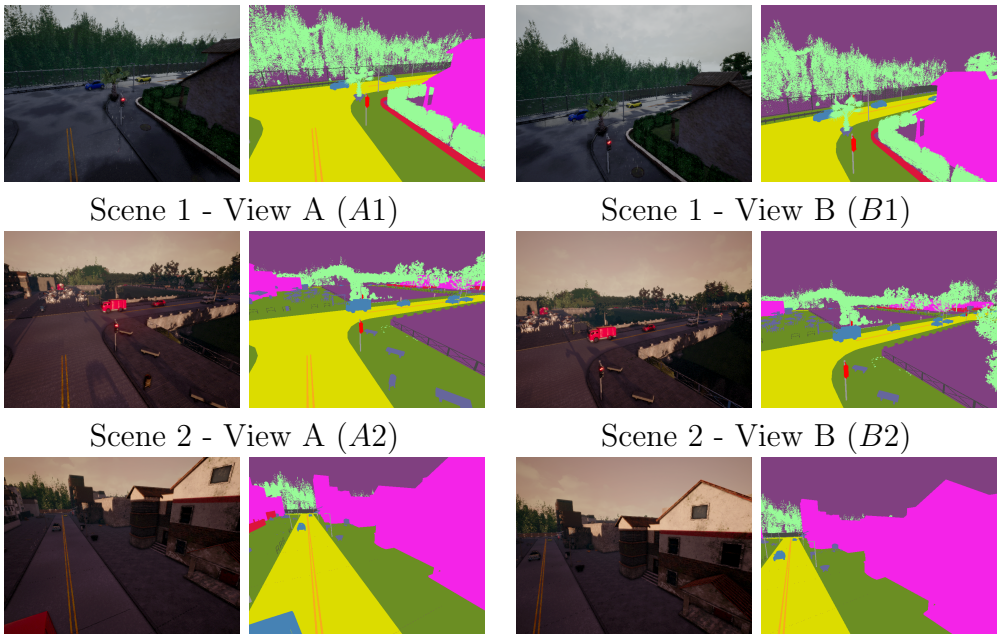
$$\mathcal{L}_{GAN}(G, F, D) = \mathbb{E}_{x \sim p_{ST}(x)}[log(D(x))] + \mathbb{E}_{x \sim p_{ST}(x)}[log(1 - D(G(F(x))))]. \quad (6.3)$$

The contribution of the reconstruction and adversarial losses to the overall performance of the proposed method are analyzed in Section 6.4.4.

Overall Loss

The overall loss used to train the whole architecture shown in Figure 6.2 is defined as the sum of the three losses discussed in previous sections:

$$\mathcal{L}(G, F, D) = \mathcal{L}_{Sem}(F) + \mathcal{L}_{Rec}(G, F) + \mathcal{L}_{GAN}(G, F, D). \quad (6.4)$$



Scene 1 - View A ($A1$) Scene 1 - View B ($B1$)

Scene 2 - View A ($A2$) Scene 2 - View B ($B2$)

Scene 3 - View A ($A3$) Scene 3 - View B ($B3$)

Figure 6.3: Sample images and ground truth segmentation masks from the 6 data subsets.

6.2 Datasets

We used CARLA Urban Driving Simulator [70] to generate a synthetic dataset suitable for the experiments. Since the simulator is targeted to the collection of datasets for autonomous driving, images can only be collected from moving cars. To overcome this limitation, we collect images from fixed positions placing the car at a predefined position and setting up multiple cameras at different altitudes, pitch, yaw and roll angles relatively to the car. The car does not move during the simulation in order to obtain a fixed point of view, while other objects (i.e. other vehicles and pedestrians) are allowed to traverse the scene. The cameras are placed making sure that the car does not appear in the scene. Using the aforementioned procedure, we collected three episodes in three different scene contexts of the virtual city, which are referred to as "Scene 1", "Scene 2" and "Scene 3". Images from each scene context have been collected from two different points of view presenting scene overlap. The two views are referred to as "View A" and "View B". The dataset consists of 6 image subsets denoted as "XY", where X represent the view and Y represents the scene (e.g., $A1$, $B2$, etc.). Each subset is split into a training set and test set. Figure 6.3 reports some examples from the 6 subsets.

We collected $5,000$ frames at 1 fps for each episode-view pair. The dataset contains $30,000$ frames in total. Each image has a resolution of 800×600 pixels. The time of the day and weather has been generated randomly for each episode. Each image of the dataset is associated to a ground truth semantic segmentation mask produced by the simulator. The 13 annotated classes are: *buildings*, *fences*, *other*, *pedestrians*, *poles*, *road-lines*, *roads*, *sidewalks*, *vegetation*, *vehicles*, *walls*, *traffic signs* and *none*.

The collected dataset allows us to consider two different kinds of source-target domain pairs: 1) point of view adaptation pairs, composed by two subsets from the same scene context but with different views (e.g., $A1 - B2$, $A2 - B2$, etc.), 2) scene adaptation pairs, composed by two subsets belonging to different scene contexts (e.g., $A1 - A2$, $A2 - A3$, etc.). The dataset is publicly available for research purposes at http://iplab.dmi.unict.it/ParkSmartSceneAdaptation/pr.html.

6.3 Experimental Settings

We compare the proposed algorithm with respect to the following methods:

No Adaptation (NA) a baseline obtained training PSPNet on the Source Domain and testing it on the Target Domain, without adaptation;

Fine Tuning (FT) a strong baseline obtained performing the fine-tune procedure, i.e., training PSPNet directly on the Target Domain using ground truth annotations. It should be noted that, making use of ground truth annotations, the FT baseline denotes in fact the ideal performance that scene adaptation techniques should be able to achieve;

ASEGNET [48] the domain adaptation method proposed in [48]. We use the official implementation provided by the authors for the experiments;

LSD [47] the domain adaptation method proposed in [47]. We use the official implementation provided by the authors for the experiments;

Geometric Warp (WARP) this baseline assumes that an affine transformation $\mathcal{H}$ between source and target scenes exists and is known. Hence, it can be applied only in the case of point of view adaptation. The baseline works as follows. At training time, all images from the Source Domain are warped using $\mathcal{H}$ in order to match the geometry of the Target Domain. The same transformation is applied to the ground truth segmentation masks. A PSPNet model is hence trained on the warped images. The network is tested directly on the Target Domain images. It should be noted that no warp operation is needed at test time. Since the source images need to be mapped to a higher resolution in order to preserve the scale of the objects appearing in the scene, training this baseline is computationally expensive. Given the unpromising results obtained using this method and its limited applicability (i.e., to the case of point of view adaptation only), we tested the method only on a subset of the data, as it is discussed in Section 6.4.3.

We perform two experiments to assess the performance of the proposed method in the contexts of point of view adaptation and general scene adaptation. In the first set of experiments (point of view adaptation), the Source and Target Domains are

related to images acquired in the same scene from different points of view. In the second set of experiments (general scene adaptation), Source and Target Domains are related to images acquired in different, non-overlapping scenes.

All algorithms are trained for 10 epochs (all methods converged before 10 epochs in our experiments). The weights of the best epoch are selected for the evaluation. We optimize the baselines (i.e. NA and FT) using Stochastic Gradient Descent (SGD) with momentum equal to 0.9, initial learning rate equal to 0.007, weight decay equal to $(1 - i/3750)^{0.9}$ (where i is the current iteration, and 3750 is the total number of iterations) and batch size equal to 8. The proposed method is optimized using Adam, with initial learning rate equal to 0.0002 and batch size equal to 1. LSD [47] is trained using SGD with momentum equal to 0.9, initial learning equal to $1.0e^{-}5$ and weight decay coefficient equal to 0.0005. ASEGNET [48] is trained using SGD with momentum 0.9, initial learning equal to $2.5e^{-}4$ and weight decay coefficient equal to 0.0005.

All the results have been evaluated using Per Class accuracy (c_{acc}) and Mean Intersection Over Union (m_{iou}), which are standard measures for semantic segmentation. The measures are defined as follows [31]:

$$c_{acc} = \frac{1}{n_{cl}} \sum_i \frac{n_{ii}}{t_i} \tag{6.5}$$

$$m_{iou} = \frac{1}{n_{cl}} \sum_i \frac{n_{ii}}{(t_i + \sum_j n_{ji} - n_{ii})} \tag{6.6}$$

where n_{ij} indicates the number of pixels of class i classified as belonging to class j, n_{cl} the total number of classes, and $t_i = \sum_j n_{ij}$.

6.4 Results

6.4.1 Comparison with "the state of the art"

Table 6.1 and Table 6.2 report the results of the first set of experiments aimed to study point of view adaptation. Each table reports the average results in the first row, as well as the breakdown according to the different classes. The results are averaged over the following source-target image pairs: $A1 - B1$, $A2 - B2$, $A3 - B3$, $B1 - A1$, $B2 - A2$, $B3 - A3$. Specifically, each experiment on a given source-target

Table 6.1: Results of point of view adaptation experiments - per-class accuracy. Best per-row results are reported in bold. *FT* results are reported in italic for reference. The cases in which the proposed method outperforms the *FT* strong baseline are underlined.

	NA	**ASEGNET [48]**	**LSD [47]**	**OUR**	***FT***
Average	0.53	0.52	0.59	**0.72**	*0.69*
None	0.27	0.80	0.74	**0.90**	*0.96*
Buildings	**0.96**	0.93	0.53	0.95	*0.98*
Fences	0.28	0.36	**0.58**	0.55	*0.50*
Other	0.29	0.32	0.56	**0.60**	*0.46*
Pedestrians	0.21	0.04	0.11	**0.53**	*0.21*
Poles	0.05	0.08	**0.44**	0.25	*0.22*
Road-lines	0.36	0.38	**0.55**	0.48	*0.44*
Roads	**0.96**	0.88	0.59	0.93	*0.98*
Sidewalks	**0.97**	0.88	0.73	0.93	*0.98*
Vegetation	**0.89**	0.72	0.77	0.82	*0.85*
Vehicles	0.66	0.48	0.45	**0.94**	*0.84*
Walls	0.75	0.83	**0.88**	0.82	*0.88*
T. Signs	0.20	0.13	**0.78**	0.68	*0.73*

$X-Y$ pair consists in training the compared method using labeled images from the training set of X and unlabeled images from the training set of Y and test it on the test set of Y.

As can be noted, the proposed method outperforms with a good margin all competitors (in average) considering both per-class accuracy (Table 6.1) and mean intersection over union (Table 6.2). Interestingly, the proposed method also outperforms the *FT* strong baseline in the case of per-class accuracy (Table 6.1). The per-class scores reported in the tables, show that the main advantage of the proposed method consists in effectively preventing over-fitting improving the segmentation results for classes containing small objects such as "Fences", "Pedestrians", "Vehicles" and "T. Signs". For instance, the proposed method scores a per-class accuracy (Table 6.1) of 0.53 for the segmentation of pedestrians, while the No Adaptation baselines (NA) scores only 0.21. Similarly, the proposed method scores a per-class accuracy of 0.68 in the case of traffic signs versus 0.20 scored by the NA baseline. Similar gains can be observed for the mean intersection over union measure (Table 6.2).

Table 6.3 and Table 6.4 summarize the results related to the second set of experiments aimed at assessing the general scene adaptation capabilities of the proposed

Table 6.2: Results of point of view adaptation experiments - mean intersection over union. Best per-row results are reported in bold. *FT* results are reported in italic for reference. The cases in which the proposed method outperforms the *FT* strong baseline are underlined

	NA	**ASEGNET [48]**	**LSD [47]**	**OUR**	***FT***
Average	0.32	0.37	0.34	**0.57**	*0.64*
None	0.26	0.75	0.48	**0.85**	*0.91*
Buildings	0.74	0.87	0.49	**0.93**	*0.98*
Fences	0.13	0.15	0.33	**0.42**	*0.42*
Other	0.22	0.10	0.25	**0.50**	*0.42*
Pedestrians	0.09	0.03	0.09	**0.13**	*0.19*
Poles	0.04	0.02	0.17	**0.20**	*0.17*
Road-lines	0.20	0.03	0.10	**0.36**	*0.34*
Roads	0.55	0.82	0.41	**0.86**	*0.95*
Sidewalks	0.76	0.66	0.48	**0.88**	*0.92*
Vegetation	0.52	0.49	0.64	**0.71**	*0.74*
Vehicles	0.23	**0.37**	0.33	0.26	*0.78*
Walls	0.28	0.52	0.31	**0.71**	*0.83*
T. Signs	0.16	0.03	0.39	**0.63**	*0.70*

method. In this case, the results are averaged over the following source-target image pairs: $A1-A2$, $A1-A3$, $A2-A1$, $A2-A3$, $A3-A1$, $A3-A2$. The proposed method outperforms all competitors in the case of mean intersection over union (Table 6.4) and achieves state-of-the-art results in the case of per-class accuracy (Table 6.3). The results highlight that general scene adaptation is much more difficult than point of view adaptation. Specifically, the proposed method tends to outperform the *FT* strong baseline less often than in the previous set of experiments. Due to the radical change of the image layout due to the very different scenes, the proposed method is not always able to accurately segment small objects such as pedestrians and traffic lines, albeit it generally improves over the NA baseline.

Figure 6.4 and 6.5 report some qualitative examples of the compared methods. The examples in Figure 6.4 confirm the observations made on the basis of the quantitative results, i.e., the proposed method allows to recover small details (e.g., traffic signs), especially when compared with respect to the No Adaptation (NA) baseline. The effect is more substantial int he case of scene adaptation (Figure 6.5), where the proposed method allows to reduce over-fitting to the source domain.

Table 6.3: Results of scene adaptation experiments - per-class accuracy. Best per-row results are reported in bold. *FT* results are reported in italic for reference. The cases in which the proposed method outperforms the *FT* strong baseline are underlined.

	NA	**ASEGNET [48]**	**LSD [47]**	**OUR**	***FT***
Average	0.27	**0.37**	0.32	0.34	*0.68*
None	0.53	0.54	0.39	**0.69**	*0.23*
Buildings	0.09	0.18	0.17	**0.25**	*0.99*
Fences	0.07	0.03	**0.25**	0.01	*0.52*
Other	0.02	**0.07**	0.02	0.00	*0.70*
Pedestrians	0.05	0.01	0.03	**0.13**	*0.38*
Poles	0.03	0.11	**0.25**	0.03	*0.25*
Road-lines	0.13	**0.66**	0.49	0.52	*0.43*
Roads	**0.84**	0.78	0.75	0.77	*0.99*
Sidewalks	0.67	**0.69**	0.67	0.50	*0.98*
Vegetation	0.41	0.22	0.39	**0.49**	*0.94*
Vehicles	0.61	0.57	0.35	**0.81**	*0.89*
Walls	**0.00**	**0.00**	**0.00**	**0.00**	*0.94*
T. Signs	0.04	**0.99**	0.73	0.18	*0.65*

Table 6.4: Results of scene adaptation experiments - mean intersection over union. Best per-row results are reported in bold. *FT* results are reported in italic for reference. The cases in which the proposed method outperforms the *FT* strong baseline are underlined.

	NA	**ASEGNET [48]**	**LSD [47]**	**OUR**	***FT***
Average	0.19	0.18	0.20	**0.22**	*0.47*
None	0.30	0.46	0.17	**0.50**	*0.22*
Buildings	0.07	0.09	0.25	**0.25**	*0.90*
Fences	**0.06**	0.01	0.02	0.01	*0.21*
Other	**0.02**	0.01	0.01	0.00	*0.40*
Pedestrians	**0.03**	0.01	**0.03**	0.02	*0.15*
Poles	0.01	0.01	**0.08**	0.02	*0.25*
Road-lines	0.13	0.07	0.27	**0.40**	*0.33*
Roads	0.62	**0.71**	0.65	0.59	*0.88*
Sidewalks	**0.45**	0.44	0.40	0.43	*0.88*
Vegetation	0.28	0.10	0.23	**0.33**	*0.54*
Vehicles	0.40	**0.48**	0.25	0.21	*0.71*
Walls	**0.00**	**0.00**	**0.00**	**0.00**	*0.71*
T. Signs	0.04	0.00	**0.31**	0.16	*0.07*

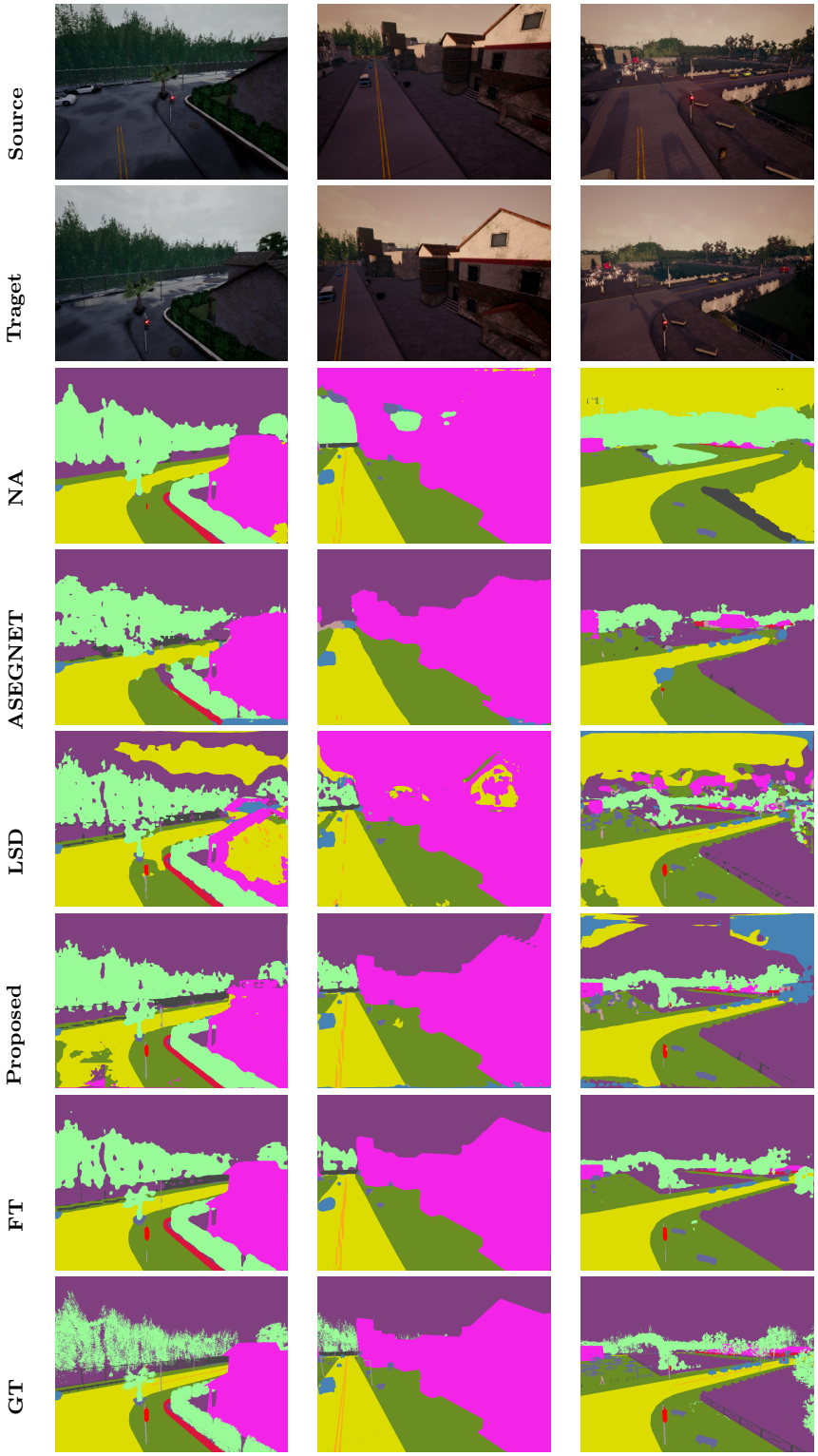


Figure 6.4: Qualitative comparisons of view adaptation with respect to the considered methods.

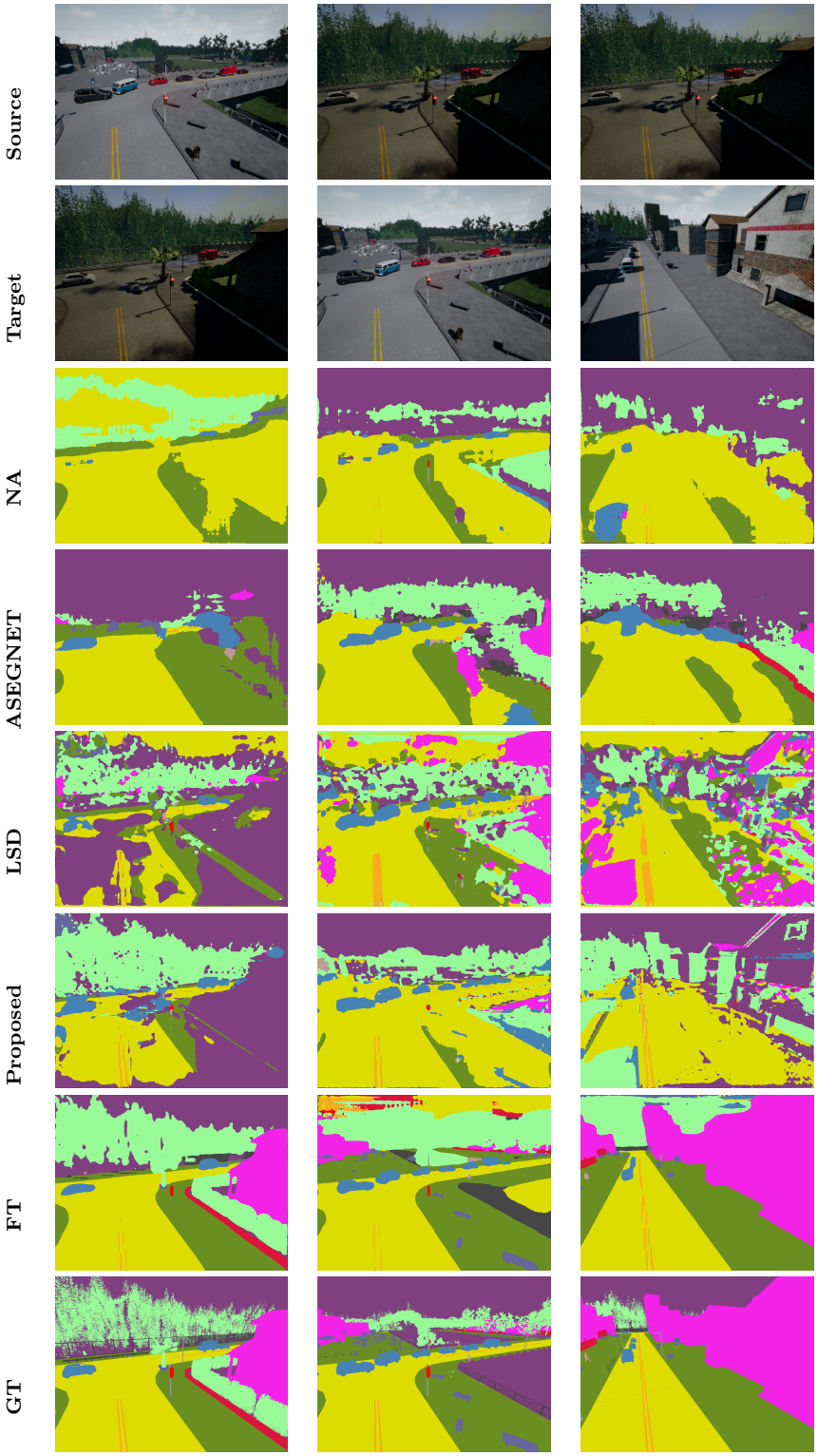


Figure 6.5: Qualitative comparisons of scene adaptation with respect to the considered methods.

Table 6.5: Improvement on the source domain.

Source	Target	Method	Measure	Test Results
$A1$	-	No Adaptation	Per Class Accuracy	0.70
$A1$	-	No Adaptation	Mean IoU	0.64
$A1$	$B1$	Proposed	Per Class Accuracy	0.74
$A1$	$B1$	Proposed	Mean IoU	0.65
$A1$	$A2$	Proposed	Per Class Accuracy	0.75
$A1$	$A2$	Proposed	Mean IoU	0.66

Table 6.6: Results of the WARP baseline for the $A1$-$B1$ pair.

	Per Class Accuracy	MIoU
NA	0.47	0.43
WARP	0.58	0.46
Proposed	**0.75**	**0.62**
FT	*0.72*	*0.66*

6.4.2 Improvement of Performance on the Source Domain

We performed additional experiments in order to assess whether the proposed method allows to improve segmentation results on the source domain thanks to the target domain images seen during the training phase. Specifically, to assess generalization in the case of point of view adaptation, we trained the proposed method on the training set of the source-target pair $A1 - B1$ and tested it on the test set of the source domain set $A1$. Similarly, for scene adaptation, we trained the proposed method on the training sets of the source-target pair $A1 - A2$ and tested it on the test set of the source domain $A1$. Table 6.5 summarizes the results and observed gains with respect to the NA baseline. As can be noted, the proposed method allows to improve performance on the source domain. For instance, the proposed method improves per-class accuracy on the source domain by 5% in the case of scene adaptation and 4% in the case of point of view adaptation. This suggests a regularizing effect induced by the use of unlabeled images from the target domain.

6.4.3 Geometric Warp Baseline

In this section, we report additional results related to the WARP baseline. Table 6.6 compares the results obtained using the WARP baselines with respect to NA, the

Table 6.7: Results of the ablation study.

Loss	Adaptation	Per Class Accuracy	MIoU
$\mathcal{L}_{SemS}(F) + \mathcal{L}_{Rec}(G, F)$	view	**0.77**	**0.67**
$\mathcal{L}_{SemS}(F) + \mathcal{L}_{GAN}(G, F, D)$	view	0.71	0.55
$\mathcal{L}_{SemS}(F) + \mathcal{L}_{Rec}(G, F) + \mathcal{L}_{GAN}(G, F, D)$	view	0.75	0.62
$\mathcal{L}_{SemS}(F) + \mathcal{L}_{Rec}(G, F)$	scene	0.33	0.20
$\mathcal{L}_{SemS}(F) + \mathcal{L}_{GAN}(G, F, D)$	scene	0.38	**0.24**
$\mathcal{L}_{SemS}(F) + \mathcal{L}_{Rec}(G, F) + \mathcal{L}_{GAN}(G, F, D)$	scene	**0.39**	**0.24**

proposed method and the *FT* strong baseline on the A1-B1 data subset. As can be noted, the WARP baseline allows to obtain only marginal improvements over the *NA* baseline. Given the unpromising results obtained on a subset of the dataset, the computational expensiveness of the method, and its applicability only to the case of point of view adaptation, we did not perform the experiments on the whole dataset.

6.4.4 Ablation Study

To assess the contribution of the reconstruction and adversarial losses described in Section 6.1.2 and Section 6.1.2, we tested three versions of the proposed method on the $A1-B1$ pair for view adaptation and on the $A1-A2$ for scene adaptation. Each considered version of the proposed method is trained using different combinations of loss functions as it is shown in Table 6.7.

As can be noted, using both the reconstruction and adversarial loss is better for general purpose adaptation in both scenarios. On the other side, for simple view adaptation, using only the reconstruction loss leads to better results. Indeed, we can observe a boost of 2% in per-class accuracy and a 5% in mean intersection over union.

The results highlight the importance of combining both losses to obtain good results for general scene adaptation, while the method can be simplified when only point of view adaptation needs to be obtained.

6.5 Discussions

In this chapter, we have proposed a method to perform scene and point of view adaptation using adversarial learning. The method improves the generalization of a semantic segmentation network by enforcing the reconstruction of the input images

from the generated semantic segmentation class scores. Experiments show that the proposed method greatly reduces over-fitting in both point of view and scene adaptation and outperform baselines and other state-of-the-art methods.

Chapter 7

Conclusions

We built a strong background over the domain and over the deep learning methods suitable to be used to solve the problem using classical supervised approaches. Through an analysis of the domain it was easy to understand that, to build a proper dataset for training, there were several variabilities to be considered such as: camera view, shapes of the parking spaces, and other classic ones such as background, light, deformation, weather, etc.

A first attempt to decrease labeling effort was to use a Semi-Supervised approach but results shown that the supervised approach with fine-tuning, even with really few data, outperforms pseudo-label method. Such evidence suggested us to try other methods in order to make deployment scalable. We focused our attention on methods based on counting cars and parking spots, results shown that when the geometry of the scene is known, binary classification methods are always to be preferred. When the configuration of a parking lot is not known, method based on image segmentation are to be preferred over methods based on object detection. After such result we moved our attention to a full knowledge of the scene through Semantic Segmentation and Domain Adaptation techniques based on Generative Adversarial Networks in order to find a viable way to reach good trade-off between Semantic Segmentation accuracy and labeling effort. The proposed method greatly reduces over-fitting in both point of view and scene adaptation and outperform baselines and other state-of-the-art methods.

7.1 Future Directions

The next step should go through the acquisition of a real world dataset in order to improve the current system as well testing systems based on Multi-task Learning [68] which has showed several advantages, Instance Semantic Segmentation [68, 74] trying to achieve the best of the two worlds: detection and segmentation, Self-Supervised Learning [75, 76, 77] and Meta-Learning [78] which, currently represent two good candidates to tackle the curse of data labeling effort reduction.

Bibliography

[1] J. M. Shapiro. "Smart cities: quality of life, productivity, and the growth effects of human capital". In: *The review of economics and statistics* 88.2 (2006), pp. 324–335.

[2] L. Atzori, A. Iera, and G. Morabito. "The internet of things: A survey". In: *Computer networks* 54.15 (2010), pp. 2787–2805.

[3] International Telecommunication Union. *ITU-T Y.4000/Y.2060 – Overview of the Internet of Things*. 2012.

[4] A. Ahmed and E. Ahmed. "A survey on mobile edge computing". In: *Intelligent Systems and Control (ISCO), 2016 10th International Conference on*. IEEE. 2016, pp. 1–8.

[5] J.-F. Raymond. "Traffic analysis: Protocols, attacks, design issues, and open problems". In: *Designing Privacy Enhancing Technologies*. 2001, pp. 10–29.

[6] S. Battiato, G. M. Farinella, G. Gallo, and O. Giudice. "On-board monitoring system for road traffic safety analysis". In: *Computers in Industry* 98 (2018), pp. 208 –217.

[7] S. Battiato, G. M. Farinella, A. Furnari, G. Puglisi, A. Snijders, and J. Spiekstra. "An integrated system for vehicle tracking and classification". In: *Expert Systems with Applications* 42.21 (2015), pp. 7263–7275.

[8] Q. Wu, C. c. Huang, S. y. Wang, W. Chiu, and T. Chen. "Robust parking space detection considering inter-space correlation". In: *Multimedia and Expo, 2007 IEEE International Conference on*. IEEE. 2007, pp. 659–662.

[9] P. R. de Almeida, L. S. Oliveira, A. S. Britto, E. J. Silva, and A. L. Koerich. "PKLot–A robust dataset for parking lot classification". In: *Expert Systems with Applications* 42.11 (2015), pp. 4937–4949.

[10] T. Ojala, M. Pietikäinen, and T. Mäenpää. "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 24.7 (2002), pp. 971–987.

[11] V. Ojansivu and J. Heikkilä. "Blur insensitive texture classification using local phase quantization". In: *Image and signal processing*. Springer, 2008, pp. 236–243.

[12] G. Amato, F. Carrara, F. Falchi, C. Gennaro, C. Meghini, and C. Vairo. "Deep learning for decentralized parking lot occupancy detection". In: *Expert Systems with Applications* 72.15 (2017), pp. 327–334.

[13] R Yusnita, F. Norbaya, and N. Basharuddin. "Intelligent Parking Space Detection System Based on Image Processing". In: *International Journal of Innovation, Management and Technology* 3.3 (2012), p. 232.

[14] L. L. Ng and H. S. Chua. "Vision-based activities recognition by trajectory analysis for parking lot surveillance". In: *International Conference on Circuits and Systems*. 2012, pp. 137–142.

[15] E. Màrmol and X. Sevillano. "QuickSpot: a video analytics solution for on-street vacant parking spot detection". In: *Multimedia Tools and Applications* 75.24 (2016), pp. 17711–17743. ISSN: 1573-7721. DOI: 10.1007/s11042-016-3773-8. URL: https://doi.org/10.1007/s11042-016-3773-8.

[16] Y. LeCun, Y. Bengio, and G. Hinton. "Deep learning". In: *Nature* 521 (May 2015), 436 EP –.

[17] G. Cybenko. "Approximations by superpositions of a sigmoidal function". In: *Mathematics of Control, Signals and Systems* 2 (1989), pp. 183–192.

[18] A. Krizhevsky, I. Sutskever, and G. E. Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[19] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[20] V. Nair and G. E. Hinton. "Rectified linear units improve restricted boltzmann machines". In: *Proceedings of the 27th international conference on machine learning (ICML-10)*. 2010, pp. 807–814.

[21] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. "Improving neural networks by preventing co-adaptation of feature detectors". In: *CoRR* abs/1207.0580 (2012). arXiv: 1207.0580. URL: http://arxiv.org/abs/1207.0580.

[22] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. "Going deeper with convolutions". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 1–9.

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

[24] K. He, X. Zhang, S. Ren, and J. Sun. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[25] C. Desai, D. Ramanan, and C. C. Fowlkes. "Discriminative models for multi-class object layout". In: *International journal of computer vision* 95.1 (2011), pp. 1–12.

[26] S. Ren, K. He, R. Girshick, and J. Sun. "Faster R-CNN: Towards real-time object detection with region proposal networks". In: *Advances in neural information processing systems*. 2015, pp. 91–99.

[27] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi. "You Only Look Once: Unified, Real-Time Object Detection". In: *CoRR* abs/1506.02640 (2015). arXiv: 1506.02640. URL: http://arxiv.org/abs/1506.02640.

[28] X. He, R. S. Zemel, and M. A. Carreira-Perpinan. "Multiscale conditional random fields for image labeling". In: *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.* Vol. 2. 2004, pp. II–II. DOI: 10.1109/CVPR.2004.1315232.

[29] J. Shotton, J. Winn, C. Rother, and A. Criminisi. "TextonBoost: Joint Appearance, Shape and Context Modeling for Multi-class Object Recognition and Segmentation". In: *Computer Vision – ECCV 2006*. Ed. by A. Leonardis, H. Bischof, and A. Pinz. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 1–15.

[30] V. Badrinarayanan, A. Kendall, and R. Cipolla. "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2017).

[31] J. Long, E. Shelhamer, and T. Darrell. "Fully convolutional networks for semantic segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3431–3440.

[32] L. C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. "DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40.4 (2018), pp. 834–848.

[33] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. "Pyramid scene parsing network". In: *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 2881–2890.

[34] S. E. Fahlman, G. E. Hinton, and T. J. Sejnowski. "Massively parallel architectures for Al: NETL, Thistle, and Boltzmann machines". In: *National Conference on Artificial Intelligence, AAAI*. 1983.

[35] D. P. Kingma and M. Welling. "Auto-encoding variational bayes". In: *arXiv preprint arXiv:1312.6114* (2013).

[36] B. J. Frey, G. E. Hinton, and P. Dayan. "Does the wake-sleep algorithm produce good density estimators?" In: *Advances in neural information processing systems*. 1996, pp. 661–667.

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

[38] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, et al. "Photo-realistic single image super-resolution using a generative adversarial network". In: *arXiv preprint* (2016).

[39] W. Lotter, G. Kreiman, and D. Cox. "Deep predictive coding networks for video prediction and unsupervised learning". In: *arXiv preprint arXiv:1605.08104* (2016).

[40] J.-Y. Zhu, P. Krähenbühl, E. Shechtman, and A. A. Efros. "Generative Visual Manipulation on the Natural Image Manifold". In: *Proceedings of European Conference on Computer Vision (ECCV)*. 2016.

[41] I. Goodfellow. "NIPS 2016 tutorial: Generative adversarial networks". In: *arXiv preprint arXiv:1701.00160* (2016).

[42] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. "Image-To-Image Translation With Conditional Adversarial Networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 1125–1134.

[43] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks". In: *Computer Vision (ICCV), 2017 IEEE International Conference on*. 2017.

[44] S. J. Pan and Q. Yang. "A Survey on Transfer Learning". In: *IEEE Transactions on Knowledge and Data Engineering* 22.10 (2010), pp. 1345–1359. ISSN: 1041-4347. DOI: 10.1109/TKDE.2009.191.

[45] J. Hoffman, D. Wang, F. Yu, and T. Darrell. "Fcns in the wild: Pixel-level adversarial and constraint-based adaptation". In: *arXiv preprint arXiv:1612.02649* (2016).

[46] J. Hoffman, E. Tzeng, T. Park, J.-Y. Zhu, P. Isola, K. Saenko, A. Efros, and T. Darrell. "CyCADA: Cycle-Consistent Adversarial Domain Adaptation". In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by J. Dy and A. Krause. Vol. 80. Proceedings of Machine Learning Research. Stockholmsmässan, Stockholm Sweden: PMLR, 2018, pp. 1994–2003.

[47] S. Sankaranarayanan, Y. Balaji, A. Jain, S. Nam Lim, and R. Chellappa. "Learning From Synthetic Data: Addressing Domain Shift for Semantic Segmentation". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[48] Y.-H. Tsai, W.-C. Hung, S. Schulter, K. Sohn, M.-H. Yang, and M. Chandraker. "Learning to Adapt Structured Output Space for Semantic Segmentation". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[49] P. R. de Almeida, L. S. Oliveira, A. S. B. Jr., E. J. S. Jr., and A. L. Koerich. "PKLot – A robust dataset for parking lot classification". In: *Expert Systems with Applications* 42.11 (2015), pp. 4937 –4949.

[50] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. "Caffe: Convolutional Architecture for Fast Feature Embedding". In: *arXiv preprint arXiv:1408.5093* (2014).

[51] X. Zhu and A. B. Goldberg. "Introduction to semi-supervised learning". In: *Synthesis lectures on artificial intelligence and machine learning* 3.1 (2009), pp. 1–130.

[52] D.-H. Lee. "Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks". In: *Workshop on Challenges in Representation Learning, ICML*. Vol. 3. 2013.

[53] A. B. Chan, Z. S.-J. Liang, and N. Vasconcelos. "Privacy preserving crowd monitoring: Counting people without people models or tracking". In: *Conference on Computer Vision and Pattern Recognition*. IEEE, 2008, pp. 1–7.

[54] S. Chen, A. Fern, and S. Todorovic. "Person count localization in videos from noisy foreground and detections". In: *Conference on Computer Vision and Pattern Recognition*. IEEE, 2015, pp. 1364–1372.

[55] M. Li, Z. Zhang, K. Huang, and T. Tan. "Estimating the number of people in crowded scenes by MID based foreground segmentation and head-shoulder detection". In: *International Conference on Pattern Recognition*. 2008, pp. 1–4.

[56] V. Lempitsky and A. Zisserman. "Learning to Count Objects in Images". In: *Advances in Neural Information Processing Systems*. 2010, pp. 1324–1332.

[57] C. Zhang, H. Li, X. Wang, and X. Yang. "Cross-scene crowd counting via deep convolutional neural networks". In: *Conference on Computer Vision and Pattern Recognition*. IEEE, 2015, pp. 833–841.

[58] A. Ferrari, S. Lombardi, and A. Signoroni. "Bacterial colony counting with Convolutional Neural Networks in Digital Microbiology Imaging". In: *Pattern Recognition* 61 (2017), pp. 629–640.

[59] C. Arteta, V. Lempitsky, and A. Zisserman. "Counting in the Wild". In: *European Conference on Computer Vision*. 2016, pp. 483–498.

[60] C. C. Loy, K. Chen, S. Gong, and T. Xiang. "Crowd Counting and Profiling: Methodology and Evaluation". In: *Modeling, Simulation and Visual Analysis of Crowds: A Multidisciplinary Perspective*. New York, NY: Springer, 2013, pp. 347–382.

[61] V Rabaud and S Belongie. "Counting Crowded Moving Objects". In: *Conference on Computer Vision and Pattern Recognition*. IEEE, 2006, pp. 705–711.

[62] C. Arteta, V. Lempitsky, J. A. Noble, and A. Zisserman. "Interactive Object Counting". In: *European Conference on Computer Vision*. 2014, pp. 1–15.

[63] L. Fiaschi, U. Koethe, R. Nair, and F. A. Hamprecht. "Learning to count with regression forest and structured labels". In: *International Conference on Pattern Recognition*. 2012, pp. 2685–2688.

[64] Google. *Google Maps*. https://www.google.it/maps/@37.5264537,15.0741852,230m/data=!3m1!1e3?hl=en. [Online; accessed 12-March-2017]. 2018.

[65] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision (IJCV)* 115.3 (2015), pp. 211–252.

[66] V. Badrinarayanan, A. Kendall, and R. Cipolla. "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation". In: *Transactions on Pattern Analysis and Machine Intelligence* 39.12 (2017), pp. 2481–2495.

[67] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. "The pascal visual object classes (voc) challenge". In: *International journal of computer vision* 88.2 (2010), pp. 303–338.

[68] K. He, G. Gkioxari, P. Dollár, and R. Girshick. "Mask R-CNN". In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 2980–2988.

[69] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell. "Adversarial Discriminative Domain Adaptation". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 2962–2971.

[70] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. "CARLA: An Open Urban Driving Simulator". In: *Proceedings of the 1st Annual Conference on Robot Learning*. 2017, pp. 1–16.

[71] J. Johnson, A. Alahi, and L. Fei-Fei. "Perceptual losses for real-time style transfer and super-resolution". In: *European Conference on Computer Vision*. Springer. 2016, pp. 694–711.

[72] C. Li and M. Wand. "Precomputed real-time texture synthesis with markovian generative adversarial networks". In: *European Conference on Computer Vision*. Springer. 2016, pp. 702–716.

[73] M. Mathieu, C. Couprie, and Y. LeCun. "Deep multi-scale video prediction beyond mean square error". In: *arXiv preprint arXiv:1511.05440* (2015).

[74] B. Hariharan, P. Arbelaez, R. Girshick, and J. Malik. "Object instance segmentation and fine-grained localization using hypercolumns". In: *IEEE Transactions on Pattern Analysis & Machine Intelligence* 4 (2017), pp. 627–639.

[75] P. Agrawal, J. Carreira, and J. Malik. "Learning to see by moving". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 37–45.

[76] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros. "Context encoders: Feature learning by inpainting". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 2536–2544.

[77] M. Noroozi, A. Vinjimoor, P. Favaro, and H. Pirsiavash. "Boosting Self-Supervised Learning via Knowledge Transfer". In: *arXiv preprint arXiv:1805.00385* (2018).

[78] M. Ren, E. Triantafillou, S. Ravi, J. Snell, K. Swersky, J. B. Tenenbaum, H. Larochelle, and R. S. Zemel. "Meta-Learning for Semi-Supervised Few-Shot Classification". In: *Proceedings of 6th International Conference on Learning Representations ICLR*. 2018.