Universidad de Alcalá
COMISIÓN DE ESTUDIOS OFICIALES
DE POSGRADO Y DOCTORADO

ACTA DE EVALUACIÓN DE LA TESIS DOCTORAL

(FOR EVALUATION OF THE ACT DOCTORAL THESIS)

Año académico (*academic year*): 2018/19

DOCTORANDO (*candidate PHD*): **OÑORO RUBIO, DANIEL**
D.N.I./PASAPORTE (*Id.Passport*): ****0200X
PROGRAMA DE DOCTORADO (*Academic Committee of the Programme*): **D445-TECNOLOGÍAS DE LA INFORMACIÓN Y LAS COMUNICACIONES**
DPTO. COORDINADOR DEL PROGRAMA (*Department*): **TEORIA DE LA SEÑAL Y COMUNICACIONES**
TITULACIÓN DE DOCTOR EN (*Phd title*): **DOCTOR/A POR LA UNIVERSIDAD DE ALCALÁ**

En el día de hoy 26/06/19, reunido el tribunal de evaluación, constituido por los miembros que suscriben el presente Acta, el aspirante defendió su Tesis Doctoral **con Mención Internacional** (*In today assessment met the court, consisting of the members who signed this Act, the candidate defended his doctoral thesis with mention as International Doctorate*), elaborada bajo la dirección de (*prepared under the direction of*) ROBERTO JAVIER LOPEZ SASTRE // .

Sobre el siguiente tema (*Title of the doctoral thesis*): *LEARNING VISUAL REPRESENTATIONS WITH DEEP NEURAL NETWORKS FOR INTELLIGENT TRANSPORTATION SYSTEMS PROBLEMS*

Finalizada la defensa y discusión de la tesis, el tribunal acordó otorgar la CALIFICACIÓN GLOBAL[1] de (**no apto, aprobado, notable y sobresaliente**) (*After the defense and defense of the thesis, the court agreed to grant the GLOBAL RATING (fail, pass, good and excellent)*): SOBRESALIENTE

Alcalá de Henares, a 26 de JUNIO de 2019

Fdo. (*Signed*): JUAN SPAMPRIGO | Fdo. (*Signed*): [illegible] MATHIAS NIEPERT | Fdo. (*Signed*): PEDRO GIL JIMÉNEZ

FIRMA DEL ALUMNO (candidate's signature),

Fdo. (*Signed*): Daniel

Con fecha 24 de Julio de 2019 la Comisión Delegada de la Comisión de Estudios Oficiales de Posgrado, a la vista de los votos emitidos de manera anónima por el tribunal que ha juzgado la tesis, resuelve:

[✓ Conceder la Mención de "Cum Laude"]
No conceder la Mención de "Cum Laude"

La Secretaria de la Comisión Delegada

[illegible]

[1] La calificación podrá ser **"no apto" "aprobado" "notable" y "sobresaliente"**. El tribunal podrá otorgar la mención de "cum laude" si la calificación global es de sobresaliente y se emite en tal sentido el voto secreto positivo por unanimidad. (*The grade may be "**fail**" "**pass**" "**good**" or "**excellent**". The panel may confer the distinction of "cum laude" if the overall grade is "Excellent" and has been awarded unanimously as such after secret voting.*).

INCIDENCIAS / OBSERVACIONES:
(Incidents / Comments)

Universidad
de Alcalá
VICERRECTORADO DE INVESTIGACIÓN
Y TRANSFERENCIA

En aplicación del art. 14.7 del RD. 99/2011 y el art. 14 del Reglamento de Elaboración, Autorización y Defensa de la Tesis Doctoral, la Comisión Delegada de la Comisión de Estudios Oficiales de Posgrado y Doctorado, en sesión pública de fecha 24 de julio, procedió al escrutinio de los votos emitidos por los miembros del tribunal de la tesis defendida por ***OÑORO RUBIO, DANIEL***, el día 26 de junio de 2019, titulada, *LEARNING VISUAL REPRESENTATIONS WITH DEEP NEURAL NETWORKS FOR INTELLIGENT TRANSPORTATION SYSTEMS PROBLEMS* para determinar, si a la misma, se le concede la mención "cum laude", arrojando como resultado el voto favorable de todos los miembros del tribunal.

Por lo tanto, la Comisión de Estudios Oficiales de Posgrado y Doctorado **resuelve otorgar** a dicha tesis la

MENCIÓN "CUM LAUDE"

EL VICERRECTOR DE INVESTIGACIÓN Y TRANSFERENCIA
F. Javier de la Mata de la Mata
Documento fechado y firmado digitalmente

Copia por e-mail a:
Doctorando: OÑORO RUBIO, DANIEL
Secretario del Tribunal: PEDRO GIL JIMENEZ
Director/a de Tesis: ROBERTO JAVIER LOPEZ SASTRE //

Universidad de Alcalá

ESCUELA DE DOCTORADO
Servicio de Estudios Oficiales de Posgrado

DILIGENCIA DE DEPÓSITO DE TESIS.

Comprobado que el expediente académico de D./Dª DANIEL OÑORO RUBIO reúne los requisitos exigidos para la presentación de la Tesis, de acuerdo a la normativa vigente, y habiendo presentado la misma en formato: ☐ soporte electrónico [✓ impreso en papel], para el depósito de la misma, en el Servicio de Estudios Oficiales de Posgrado, con el nº de páginas: 120 se procede, con fecha de hoy a registrar el depósito de la tesis.

Alcalá de Henares a 11 de MARZO de 2019

UNIVERSIDAD DE ALCALÁ
SERVICIO DE ESTUDIOS OFICIALES DE POSGRADO

Aurora Juárez Abril

Fdo. El Funcionario

Universidad de Alcalá

DPTO. DE TEORÍA DE LA SEÑAL Y COMUNICACIONES
Campus Universitario
Crta. Madrid-Barcelona. Km. 33.600
28805 Alcalá de Henares
Tlf: +34 91 885 66 90
Fax: +34 91 885 66 99
dpto.teoriadelasenal@uah.es

D. **Roberto Javier López Sastre**, Profesor Titular de Universidad del Área de Conocimiento de Teoría de la Señal y Comunicaciones de la Universidad de Alcalá

CERTIFICA:

Que la tesis "**Learning visual representations with Deep Neural Networks for intelligent transportation systems problems**", presentada por D. Daniel Oñoro-Rubio, realizada en el Departamento de Teoría de la Señal y Comunicaciones bajo mi dirección, reúne méritos suficientes para optar al grado de Doctor, por lo que puede procederse a su depósito y lectura.

Alcalá de Henares, 22 de Febrero de 2019.

[illegible]

Fdo: Dr. D. Roberto Javier López Sastre

Universidad
de Alcalá

DPTO. DE TEORÍA DE LA SEÑAL Y COMUNICACIONES
Campus Universitario
Crta. Madrid-Barcelona, Km. 33.600
28805 Alcalá de Henares
Tlf: +34 91 885 66 90
Fax: +34 91 885 66 99
dpto.teoriadelasenal@uah.es

D. Daniel Oñoro-Rubio ha realizado en el Departamento de Teoría de la Señal y Comunicaciones, bajo la dirección del Doctor D. Roberto Javier López Sastre, la tesis doctoral titulada **"Learning visual representations with Deep Neural Networks for intelligent transportation systems problems"**, cumpliéndose todos los requisitos para la tramitación que conduce a su posterior lectura.

Alcalá de Henares, 22 de Febrero de 2019.

EL COORDINADOR DEL PROGRAMA DE DOCTORADO

[illegible]

Fdo: Dr. D. Sancho Salcedo Sanz

UNIVERSIDAD DE ALCALÁ

ESCUELA POLITÉCNICA SUPERIOR

DEPARTAMENTO DE TEORÍA DE LA SEÑAL Y COMUNICACIONES

"Learning visual representations with Deep Neural Networks for intelligent transportation systems problems"

Ph.D. Thesis

Author

D. Daniel Oñoro-Rubio

Director

Dr. D. Roberto Javier López Sastre

February 22, 2019

This book is dedicated to my sister, Aída.

Agradecimientos

La escritura de esta tesis conlleva el esfuerzo directo e indirecto de muchas personas. Para todas ellas son mis más sinceros agradecimientos.

El primer y más grande de los agradecimientos va apara mi tutor de tesis, Roberto J. López-Sastre. Muchas han sido las horas leyendo artículos, discutiendo ideas, escribiendo, revisando lo escrito, volviendo a escribir, y volviendo a revisar, y así hasta llegar al punto en el que esta tesis se halla. Por toda su paciencia y esfuerzo, mi más profunda gratitud.

Mis más sinceros agradecimiento para mis profesores y compañeros de la Universidad de Alcalá. En especial a todos los profesores del departamento de Teoría de la Señal y Comunicaciones, y a mi profesor y amigo José A. Medina-Merodio. Agradezco el apoyo que reciví de todo los estudiantes, y en especial de mi compañera de laboratorio, Carolina Redondo-Cabrera.

My deepest gratitude to NEC Labs Europe for choosing me for a Marie Curie scholarship. Beside that, I have discovered a group of the most talent and kind people that I have met. My special gratitude goes to Mathias Niepert and to Alberto García-Durán for all the time discussing ideas and for their moral support.

Mi más sincero agradecimiento para toda mi familia por todo su apoyo moral, pero en especial a mi hermana Aída por sus ánimos, y sobre todo a mi Madre, Gloría, por todo el esfuerzo diario que ha hecho por mí, no solo durante el desarrollo de esta tesis, sino durante toda su vida.

Abstract

This thesis tackles two major problems of the Intelligent Transportation Systems (ITS): the vehicle counting in traffic congestion scenes, and the simultaneous object detection and pose estimation in images.

For the vehicle counting problem, this thesis is first focused on the design of new deep neural networks architectures that have the ability to learn deep multi-scale representations able to perform a precise estimation of the object count in the form of density maps. It deals with the problem of the object scale introduced by the large perspective typically present in the object counting problem. In addition, with the success of the deep hourglass networks in the object counting field, this work proposes a new type of deep hourglass network with learnable self-gated short-cut connections. The proposed models are evaluated in the most commonly used benchmarks and achieve results equal to or greater than the state of the art at the time they were published.

For the second problem, the thesis offers a complete comparative study of the simultaneous object detection and pose estimation problem. The existing compromise between the object localization and the pose estimation task is exposed. A detector ideally needs a representation which is invariant to the viewpoint, while a pose estimator needs to be discriminative. Hence, we introduce three new deep neural networks architectures where the problems of the object detection and pose estimation are progressively decoupled. Moreover, the question of whether the pose should be expressed as a discrete or a continuous variable is addressed. Despite the similar performance, the results show that the continuous approaches are more sensitive to the bias of the main viewpoint of the object category. A detailed comparative analysis is performed on the two main datasets, *i.e.* PASCAL3D+ and ObjectNet3D. Competitive results are achieved by the proposed models in both datasets.

Index terms: object counting, counting by regression, pose estimation, object detection, convolutional neural networks, hourglass networks, gating units, deep-learning, machine learning, artificial intelligence.

Resumen

Esta tesis se centra en dos grandes problemas en el área de los sistemas de transportes inteligentes (STI): el conteo de vehículos en escenas de congestión de tráfico; y la detección y estimación del punto de vista, de forma simultánea, de los objetos en una escena.

Respecto al problema del conteo, este trabajo se centra primero en el diseño de arquitecturas de redes neuronales profundas que tengan la capacidad de aprender representaciones multi-escala profundas, capaces de estimar de forma precisa la cuenta de objetos, mediante mapas de densidad. Se trata también el problema de la escala de los objetos introducida por la gran perspectiva típicamente presente en el área de recuento de objetos. Además, con el éxito de las redes hourglass profundas en el campo del conteo de objetos, este trabajo propone un nuevo tipo de red hourglass profunda con conexiones de corto circuito auto-gestionadas. Los modelos propuestos se evalúan en las bases de datos públicas más utilizadas y logran los resultados iguales o superiores al estado del arte en el momento en que fueron publicadas.

Para la segunda parte, se realiza un estudio comparativo completo del problema de detección de objetos y la estimación de la pose de forma simultánea. Se expone el compromiso existente entre la localización del objeto y la estimación de su pose. Un detector necesita idealmente una representación que sea invariable al punto de vista, mientras que un estimador de poses necesita ser discriminatorio. Por lo tanto, se proponen tres nuevas arquitecturas de redes neurales profundas en las que el problema de la detección de objetos y la estimación de la pose se van desacoplando progresivamente. Además, se aborda la cuestión de si la pose debe expresarse como un valor discreto o continuo. A pesar de ofrecer un rendimiento similar, los resultados muestran que los enfoques continuos son más sensibles al sesgo del punto de vista principal de la categoría del objeto. Se realiza un análisis comparativo detallado en las dos bases de datos principales, es decir, PASCAL3D+ y ObjectNet3D. Se logran resultados competitivos con todos los modelos propuestos en ambos conjuntos de datos.

Index terms: conteo de objetos, conteo por regresión, estimación de pose, detección de objetos, redes neuronales convolucionales, redes hourglass, unidades de computación, aprendizaje profundo, aprendizaje de máquinas, inteligencia artificial.

Glossary

AP	Average Viewpoint Precision.
AVP	Average Viewpoint Precision.
CCNN	Counting Convolutional Neural Network.
CNN	Convolutional Neural Network.
DANN	Deep Artificial Neural Networks.
DPM	Discriminatively trained deformable Part Models.
FC	Fully-connected (layer).
FCNN	Fully Convolutional Neural Network.
GAME	Grid Average Mean absolute Error.
GU-Net	Gated U-Net.
ITS	Intelligent-Transportation-Systems.
KLT	Kanade-Lucas-Tomasi.
MAE	Mean Absolute Error.
MSE	Mean Squared Error.
MSD	Mean Standard Derivation.
SVM	Support Vector Machine.
SIFT	Scale-Invariant Feature Transform.
SURF	Speeded Up Robust Features.

ORB Oriented FAST and Rotated BRIEF.

PEAP Pose Estimation Average Precision.

AOS Average Orientation Similarity.

RCNN Region-based Convolutional Neural Networks.

ROI Region Of Interest.

RPN Region Proposal Network.

UGV Unnamed Ground Vehicles.

YOLO You Only Look Once (CNN model).

Contents

List of Figures

List of Tables

Chapter 1

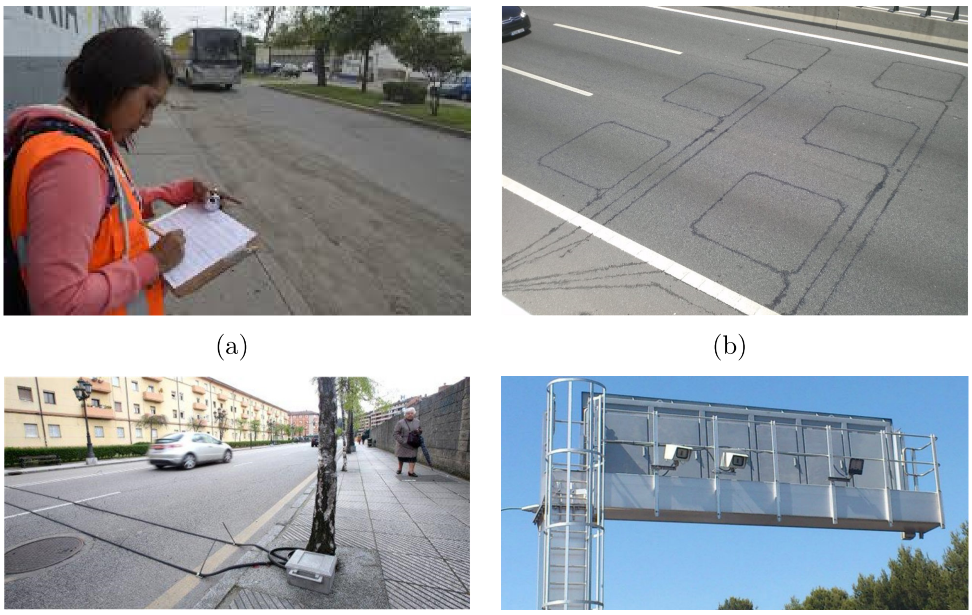
Introduction

The establishment of shared theoretical frameworks, combined with the availability of data and processing power, has yielded remarkable successes in various component tasks such as speech recognition, image classification, autonomous vehicles, machine translation, legged locomotion, and question-answering systems.
As capabilities in these areas and others cross the threshold from laboratory research to economically valuable technologies, a virtuous cycle takes hold whereby even small improvements in performance are worth large sums of money, prompting greater investments in research. There is now a broad consensus that AI research is progressing steadily, and that its impact on society is likely to increase... Because of the great potential of AI, it is important to research how to reap its benefits while avoiding potential pitfalls.

`From Research Priorities for Robust and Beneficial Artificial Intelligence, an open letter, January, 2015.`

1.1 Vehicle counting

Mobility is one of the main virtual goods of any modern society. Whether it's moving workers to labor place, transporting goods, or even enhancing tourism, mobility is one of the main pillars of the economy and people's freedom of any modern society. However, daily increasing levels of traffic

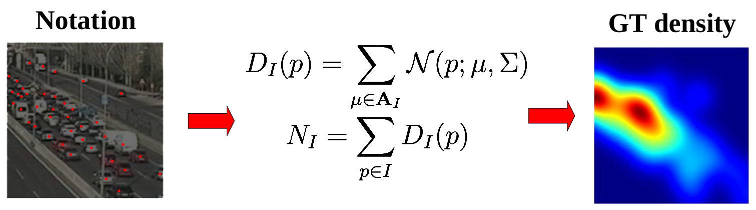

(a) (b)

(c) (d)

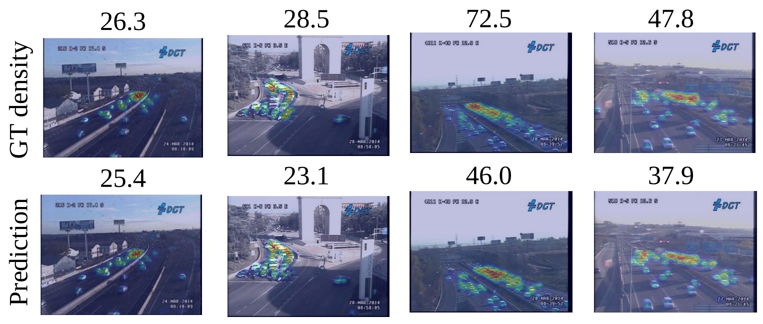
Figure 1.1: Typical methods used to monitor the traffic congestion.

congestion underscore the importance of new technological developments in the field of Intelligent Transportation Systems (ITS). One of the targets of such systems is to ensure the efficient utilization of the existing infrastructure. These technologies can help with the planning of the new and future infrastructures by offering new tools, in the continuous effort to develop accessible, safe, and sustainable transportation solutions.

Currently, the traffic congestion problem is often based on a lack of information. Every day, millions of persons take the car to commute to their work, to enjoy their vacations in a touristic place, or to enjoy their favorite social event. These actions often end up causing traffic jams, which: a) rise the accident risk, b) are costly in terms of time, and money; and c) increase pollution in big cities. Therefore, information acquisition is the first tool of a smart city towards offering solutions that improve the roads utilization and the design of new infrastructure. Data acquisition is typically performed by field operators, pressure sensors, magnetic coils or video surveillance cameras. Figure 1.1 shows some examples. Field operators are costly, they can only work for a limited time period, and cannot provide data in real time. Coils and pressure cables are limited to detecting vehicles that have passed over them, which mean they have no clue if they are in a traffic jam. Cameras are probably the most common sensor. In contrast to the pressure cables and


Notation
$D_I(p) = \sum_{\mu \in \mathbf{A}_I} \mathcal{N}(p; \mu, \Sigma)$
$N_I = \sum_{p \in I} D_I(p)$
GT density

(a)



(b)

Figure 1.2: (a) shows the typical labeling for the object counting problem. The objects are marked with a single dot on top of them. The dot annotation is converted into a density map by adding Gaussian functions centered on each dot. (b) shows some qualitative results of the proposed GU-Net model.

coils, they provide multi-purpose information that can cover large areas. For this reason, this thesis is focused on image data sources.

Image-based object counting is a very interesting challenge of the computer vision and intelligent transportation systems communities. The nature of the problem typically leads to situations where the objects are highly occluded. The camera perspective, the background variability, and the light and weather conditions are some of the factors that also directly affect the performance of the standard solutions based on object detection, segmentation, and tracking methods. Hence, developing systems that are robust to all of these conditions is an essential step towards offering automatic object counting solutions.

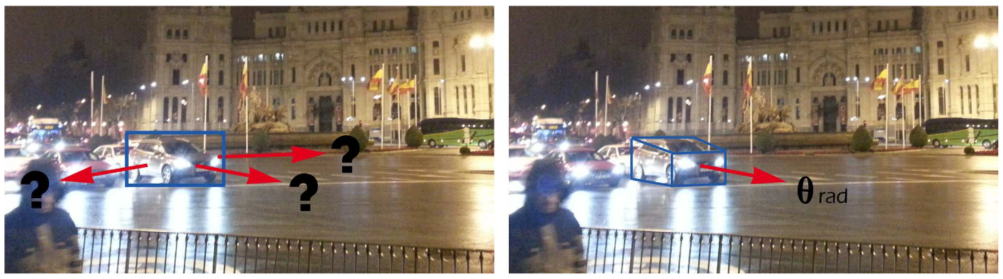
Deep Artificial Neural Networks (DANN) are a powerful modeling technique that can map an arbitrary input with a given target. The way they work is by projecting the input data into intermediate latent representations that are learned during an optimization process. Those intermediate representations can be understood as a collection of features that progressively reduce the complexity of the problem. After all the transformations, a linear classifier is powerful enough to perform the final prediction with high performance. This thesis focuses on designing novel deep learning models that learn robust visual representations to the image scale introduced by the camera perspective, the object occlusion, the cluttered background, and other artifacts that typically make the conventional techniques unfeasible.

All our networks understand the counting problem as a regression problem. Where an input image is transformed into a density map where we can perform a precise count of the objects present in the scene. Figure 1.2(a) shows the dot annotation used to mark the objects for the vehicle counting task. The dots are converted into a density map by summing up Gaussian functions centered on each dot. In Figure 1.2(b) we anticipate the type of output that our vehicle counting solutions can offer.

1.2 Object detection and pose estimation

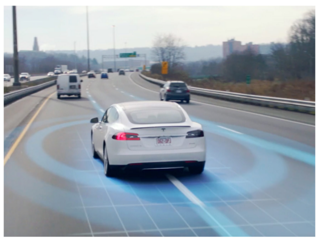
Object detection and pose estimation are core problems in computer vision and of utmost importance for the ITS and the robotics fields. The capability of understanding or inferring the 3D world through 2D images brings a wide range of possibilities. With an especial interest in ITS and smart cities, understanding the traffic flow in a highway by detecting and estimating the pose of vehicles circulating brings some interesting benefits. For instance getting information about the traffic flow can be useful for optimizing the traffic control resources of the city, see Figure 1.3(a). Getting the relative position, orientation and type or other vehicles that appears in an autonomous driving scenario is essential in the development of Unnamed Ground Vehicles (UGV), such as the Google's UGV, the semi-autonomous Tesla cars, and other prototypes that may enable the fully automatized and smart traffic of the future (see Figure 1.3(b), 1.3(c), and 1.3(d) for some samples). Estimating the relative position of certain objects that need to be manipulated by a robot is another example of an application where the object detection and pose estimation plays a key role in the systems where the robot needs to interact with the environment (see Figure 1.3(e)).

Object detection refers to the problem of localizing and recognizing certain object categories within an image. Pose estimation deals with predicting the relative position of the object with respect to the camera in the 3D space. Object detection and pose estimation from images is a longstanding and challenging problem. The major difficulty is given by the fact that the appearance of a given object class may drastically change by a lot of factors such as the intra-class variability, the viewpoint, scale, occlusions, truncation, light conditions, sensor noise, etc. Inspired by the ability of human beings in recognizing objects in a wide range of situation, pioneers in



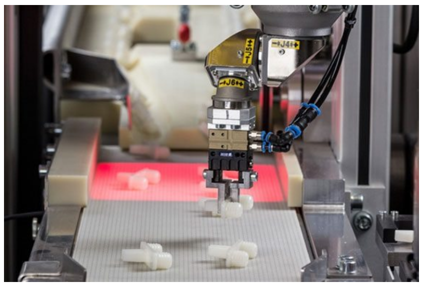
(a) Car detection and pose estimation system for smart cities (Guerrero-Gómez-Olmedo et al. 2013).

(b) Google's UGV.



(c) Tesla car.

(d) KoreaTech University's UGV.



(e) Robotic arm interacting with objects with multiple orientations.

Figure 1.3: Application examples of the object detection and pose estimation.

machine learning community aimed at building computational models that learn latent representations that are robust to the aforementioned challenge but, in addition, they need to be discriminative with respect the object class and its pose. Remarkable progress has been achieved in the recent years due to the advances in the computing hardware such as the GPUs, storage systems, the development of the Internet, and the endless effort of the machine learning, computer vision and intelligent transportation systems communities. Although, despite all these advances, the performance of a system is bounded by the biases on these datasets. There are few large datasets such as the PASCAL3D+ (Xiang et al. 2014) or the ObjectNet3D (Xiang et al. 2016) datasets that allows to establish a detailed experimental evaluation for the problems of object detection and pose estimation. However, they hold collections of natural images extracted from the Internet, hence the object categories are biased toward the most common class and towards the most common view. As an example, let consider the case of a car. It is not natural to take photos of the underneath of the car, which causes a bias that affects the performance of the models in charged of the estimation of the viewpoint.

Object representation is a crucial component of most of the object detection and pose estimation methods. Since there is a large intra-class variability, similarities between classes, occlusions and other difficulties, obtaining robust object representation is the component with the highest impact in the performance of a system. There are many methods for learning efficient representations such as in (Lowe 1999, Bay et al. 2006, Rublee et al. 2011, Schmid 2006, Girshick et al. 2014). However, all those representation methods are applied and fitted to datasets that typically hold biases towards certain object categories, and poses. Further, the object detection and the pose estimation are two different tasks that, up to a certain extent, overlap. While it is common for both to be robust to light conditions, occlusions, the background, and other aspects, the object detection needs to be invariant to the viewpoint, while the pose estimation needs to be discriminative. In addition to that, there are two different ways of understanding the problem of the pose estimation that may affect the performance of the systems: a continuous approach, or a discrete manner. In this thesis, it is analyzed how all the aforementioned aspects affect the system performance. We introduce three novel deep learning architectures, for the problem described, with a thorough experimental evaluation that allows us to validate the degree of coupling between the object localization and the pose estimation tasks.

1.3 Objectives of the thesis

This thesis aims to understand the main difficulties that affect the system's performance for two of the main problems in ITS: vehicle counting and simultaneous object detection and pose estimation.

During the first part of this thesis, it is tackled the object counting problem. The change of scale introduced by the perspective is one the main problem in image recognition, but in especial, in the object counting field. Due to the nature of the problem, the images exhibit a large change of perspective. That makes that the objects close to the camera appear significantly bigger than those close to the vanishing point of the scene. This thesis first introduces a new type of scale-aware neural network model that aims to overcome the problem of the scale. Additionally, we study the impact of the information transferred in the skip-connections of the hourglass architectures, and a new mechanism to gate the information which is forwarded is proposed.

The second part of this thesis is focused on the simultaneous object detection and pose estimation problem. This problem aims to detect object instances in 2D images and to predict their class labels and 3D poses with respect to the camera. It is a broad problem that can be tackled in several ways. Depending on the methodology, the proposed methods in the literature can be classified according to whether or not they simultaneously perform the two tasks, and whether the predicted pose is a continuous or discrete variable. This thesis studies the impact of decoupling or not the detection from the pose estimation, and the impact of a continuous pose prediction versus a discrete approach. In order to do that, three new models that implement these requirements are introduced. Providing an extensive and detailed experimental evaluation, in multiple datasets, it is observed the impact of the aforementioned approaches, but it also put into manifest the impact in the performance of the dataset biases for the different methodologies designed.

1.4 Thesis outline

This thesis is organized as follows:

- **Chapter 2** is focused on the object counting task. In this chapter, it is first introduced the object counting problem. Then, a complete review of the previous methods is provided. Next, the datasets that have been used for the evaluation are introduced. Interestingly, our experimental evaluation considers not only datasets for vehicle counting, but also for crowd counting. Then, it is introduced the proposed Counting Convolutional Neural Network (CCNN) and Hydra architectures and their experimental evaluation. These contributions appear in the first top-tier publication of this thesis (Oñoro-Rubio and López-Sastre 2016). Second, the counting U-Net and the GU-Net models (related to the second top-tier publication of this thesis (Oñoro-Rubio et al. 2018)) and their experimental evaluations are introduced. A final discussion and the conclusions are provided at the end of the chapter.

- **Chapter 3** tackles the problem of the object detection and pose estimation. First, an introduction to the problem is presented. A subsection with the related works is introduced afterward. Then, the notation and the proposed model architectures are presented. We evaluate up to three architecture designs where we gradually decouple the object localization and the viewpoint estimating tasks. After, the different factors that affect the system's performance are analyzed by an extensive experimental setup using the datasets PASCAL3D+ and ObjectNet3D. At the end of the chapter, the conclusions of the chapter are presented. As the resulting of this chapter, the top tier journal publication (Oñoro-Rubio et al. 2018) was achieved.

- **Chapter 4** summaries the work and the contributions of this thesis. The chapter finalizes with the future research lines.

Chapter 2

Object Counting

Perhaps we should all stop for a moment and focus not only on making our AI better and more successful but also on the benefit of humanity.

`Stephen Hawking.`

2.1 Introduction

Take an image of a crowded scene, or of a traffic jam. We address here the hard problem of accurately counting the objects instances in these scenarios. Developing this type of solutions covers a wide spectrum of applications:

- Improving public safety and security by automatically monitoring crowd behavior and generating alerts that can be used to avoid stampedes.
- Collecting politically unbiased statistics of protests and pilgrims.
- Systems that automatically optimize the traffic flow by precisely monitoring how the traffic congestion evolves.

Improving object counting technology will make possible the deployment of robust solutions for the aforementioned applications, and probably in the future, tragedies such as the crowd crush at the Madrid Arena during the Halloween festival Spain at 2012, might be avoided.

In this chapter, we address the problem of counting objects instances in images. Our models are able to precisely estimate the number of vehicles in a traffic congestion or to count the humans in a very crowded scene. Our first contribution is the proposal of a novel convolutional neural network solution, named Counting CNN (CCNN). Essentially, the CCNN is formulated as a regression model where the network learns how to map the appearance of the image patches to their corresponding object density maps. Our second contribution consists of a scale-aware counting model, the Hydra CNN, able to estimate object densities in very crowded scenarios where no geometric information of the scene can be provided. Hydra CNN learns a multiscale non-linear regression model which uses a pyramid of image patches extracted at multiple scales to perform the final density prediction. Our third contribution is a counting U-Net, a version of the popular model proposed by Ronneberger et al. (2015) which has been adapted for the object counting task. The fourth contribution is the Gated U-Net (GU-Net), a novel self-gated hourglass Fully Convolution Neural Network. GU-Net implements an efficient and adaptive gating mechanism in the short-cut connections that improve the information flow over the network. We report an extensive experimental evaluation, using up to four different object counting benchmarks, where the proposed solutions reported the state-of-the-art performance at the time they were published and still competitive with the most recent methods.

This chapter is organized as follows. Section 2.3 introduces the public benchmarks that are used to evaluate the proposed models. Section 2.4 formulates the notation used for object counting. Section 2.5 details and analyzes the three novel deep network architectures, named Counting CNN (CCNN), Hydra CNN, and GU-Net. In Section 2.6, a discussion between the proposed solutions is provided. Finally, Section 2.7 summarizes the contributions made in this chapter.

2.2 Related work

Significant progress has been made to count objects in images. We refer the reader to the survey of Loy et al. (2013). Following the taxonomy introduced in Loy et al. (2013), the counting algorithms can be classified into three groups: counting by detection, counting by clustering, and counting by regression.

2.2.1 Counting by detection

The most intuitive approach to tackle this problem is to scan the target image with an object detector and to count the number of positive detections, *e.g.* Guerrero-Gómez-Olmedo et al. (2015), Leibe et al. (2005), Li et al. (2009), Tuzel et al. (2008), Li et al. (2008). Traditional detectors first scan the input

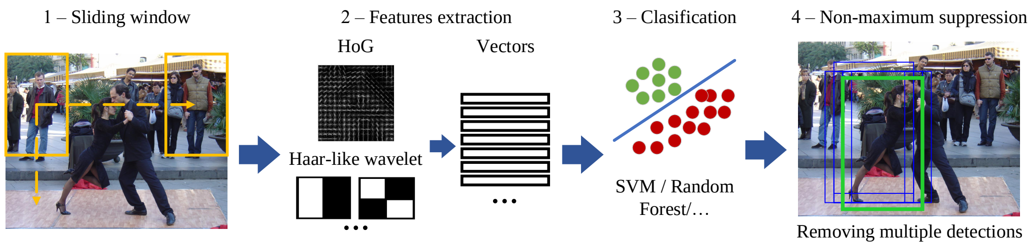


Figure 2.1: Typical object detection system based on "hand-crafted" features.

image in a sliding window fashion. From each window they extract "hand-crafted" features such as Haar wavelets (Viola and Jones 2004) or HOG (Dalal and Triggs 2005). Then a classifier, typically an SVM or a random forest, is used to distinguish the patches which correspond to the target object from the samples of the background. Finally, to filler out the multiple weak detections around the object, a non-maximum suppression filtering is applied. Figure 2.1 illustrates this process.

With the success of the deep learning in the computer vision community, the feature extraction became an automatic process which substantially boosted the performance of the object detection solutions. RCNN (Girshick et al. 2014), Fast RCNN (Girshick 2015), or YOLO (Redmon et al. 2016, Redmon and Farhadi 2017) define the current state-of-the-art. A typical Faster RCNN system starts with a deep CNN which extracts deep features. Based on those features a Region Proposal Network(RPN) identifies important regions on the input image. Then, the deep features that correspond to the generated ROIs are pooled. Another deep neural network classifies the retrieved regions, and finally a non-maximum suppression algorithm filter out the multiple detentions. Figure 2.2 depicts the typical diagram of the approaches.

Figure 2.3 shows some qualitative detections of some of the most relevant works. As could be observed, counting by detection may work well in those scenarios where the objects are sparse and can be clearly distinguished from the background and from each other. By taking a look to Figure 2.3, it can be clearly observed in (a) how the overlap between the cars, the scale, and the viewpoint, makes the detector struggle to locate each car. Figure 2.3 (b) depicts a situation where the targets can be clearly distinguished, however, it can be observed how people that appear truncated on the top of the image is ignored. (c) shows the best case scenario where the objects are clearly visible and without much distortion due to the scale or viewpoint. Due to the nature of the problem, the typical scenarios are closer to (a) rather than (b)

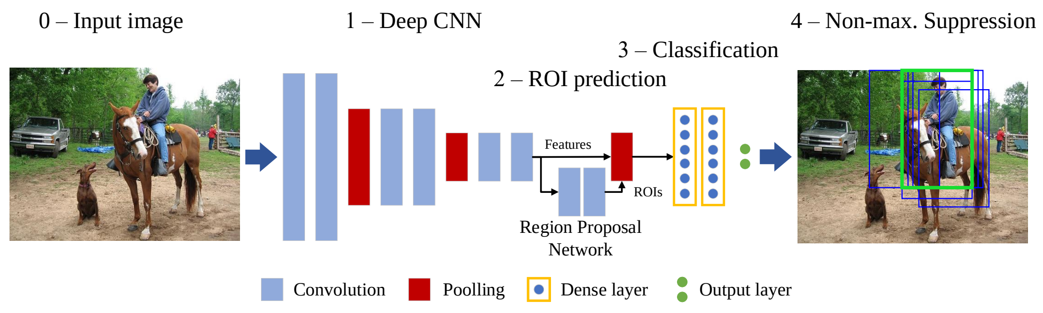


Figure 2.2: Typical faster RCNN object detection diagram.

and (c), therefore, approaches like in Guerrero-Gómez-Olmedo et al. (2015), Leibe et al. (2005), Li et al. (2009), Tuzel et al. (2008), Li et al. (2008) are expected to have a poor performance.

2.2.2 Counting by clustering

Clustering approaches rely on the assumption that the same type of object in a scene presents very similar and uniform visual features, motion fields and trajectories. Consequently, they can be grouped together. Following this line of research Rabaud and Belongie (2006) make use a Kanade-Lucas-Tomasi (KLT) (Lucas and Kanade 1981) tracker to extract a set of low-level features. Those features are grouped by a clustering algorithm and used to infer the number of people in the scene. In the work of Brostow and Cipolla (2006) they track local features and groups by using Bayesian clustering. In contrast to the counting by detection, these methods avoid the supervised learning and the explicit modeling of the appearance by assuming motion coherency. In Figure 2.4 it is shown some qualitative results collected from the original papers. These methods are restricted to video sequences, therefore false estimations may rise when the objects remain static in the scene, or when object shares common feature trajectories over time.

2.2.3 Counting by regression

In contrast with counting by detection, counting by regression solutions predict the total amount of objects of a given image, rather than trying to localize every single object instance. As illustrated in Figure 2.5, the general idea is to define a model which learns the regression function that projects the image appearance into an object density map. This allows the derivation of an estimated object density map for unseen images. Note that the

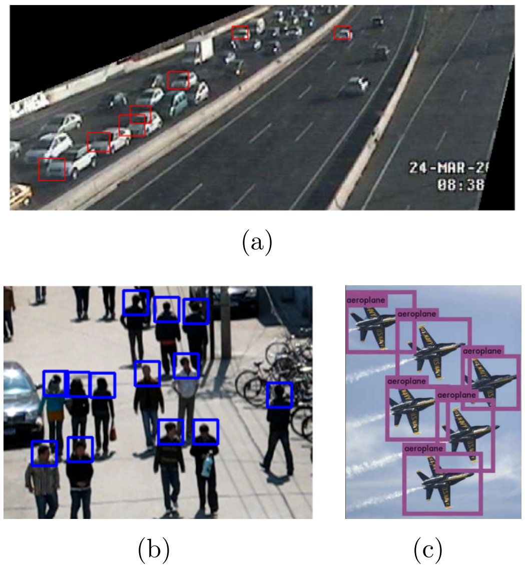


Figure 2.3: This figure shows the qualitative results of some of the most popular methods in counting by detection. The images are taken from (Guerrero-Gómez-Olmedo et al. 2015), (Li et al. 2008), and (Redmon et al. 2016) respectively.

covered applications define the typical scenarios where individual object detectors (*e.g.* Dalal and Triggs (2005), Felzenszwalb et al. (2010)) do not work reliably. The reasons are the extreme overlap of objects, the size of the instances, scene perspective, etc. Thus, approaches modeling the counting problem as one of object density estimation have been systematically defining the state-of-the-art (Arteta et al. 2014, Chan et al. 2008, Fiaschi et al. 2012, Lempitsky and Zisserman 2010, Zhang et al. 2015, 2016, Babu Sam et al. 2017, Pham et al. 2015, Ranjan et al. 2018, Cao et al. 2018, Idrees et al. 2018, Laradji et al. 2018).

Counting by regression has an extensive literature, where a special attention deserves the learning-to-count model of Lempitsky and Zisserman (2010). They introduce a counting approach, which works by learning a linear mapping from local image features to object density maps. With a successful learning, one can provide the object count by simply integrating over regions in the estimated density map. This strategy is followed also by Fiaschi et al. (2012) and by Pham et al. (2015) where a structured learning framework is applied to the random forests so as to obtain the object density map estimations. In (Arteta et al. 2014), the authors propose an interactive counting system, which simplifies the costly learning-to-count approach (Lempitsky and Zisserman 2010), proposing the use of a simple ridge regressor.

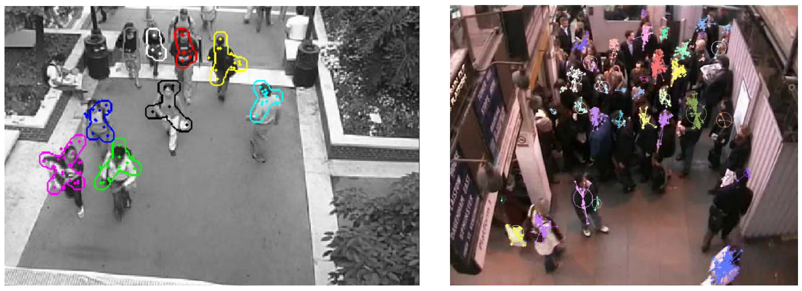


Figure 2.4: (a) and (b) show the results of clustering coherent motions using methods proposed in Rabaud and Belongie (2006) and Brostow and Cipolla (2006) respectively.

One more time, the success of the deep convolutional neural networks brought a totally new bunch of methods. The work of Zhang et al. (2015) proposes a CNN architecture to predict density maps, which is trained following a switchable learning process that uses two different loss functions. In (Zhang et al. 2016) a multi-column CNN is proposed, which stacks the feature maps generated by filters of different sizes and combine them to generate the final prediction for the count. In (Zhang et al. 2017) they proposed the first hourglass network for object counting. In their work they make use of an hourglass network as a feature extractor, then the image is divided in a grid and a regressor with a low-rank constraint is learned for each region. Li et al. (2018) propose a hourglass network based on the VGG16 architecture (Simonyan and Zisserman 2014) with dilated convolution filters. In (Laradji et al. 2018) they propose a FCNN which is trained to minimize a four termed loss function. Their solution generates a blob mask for each object instance. The total number of objects is deducted by counting the number of blobs. Idrees et al. (2018) introduce a new dataset and propose a FCNN which predicts multiple density maps where the radio of the Gaussian function placed on top of each object is progressively decreased.

In this thesis, we propose the Hydra model, which learns a multi-scale latent representation that comes from multiple scaled version of the input image. In contrast to (Zhang et al. 2016, Cao et al. 2018) where a single image is forwarded in a few columns with different filters sizes, and whose outputs are then aggregated, the presented approach takes advantage of the multiple view scales of the image and learns a specific representation for each scale that later on it is combined. In addition, the presented GU-Net improves the popular hourglass neural network architecture used in many works such as (Zhang et al. 2017, Li et al. 2018, Laradji et al. 2018) by

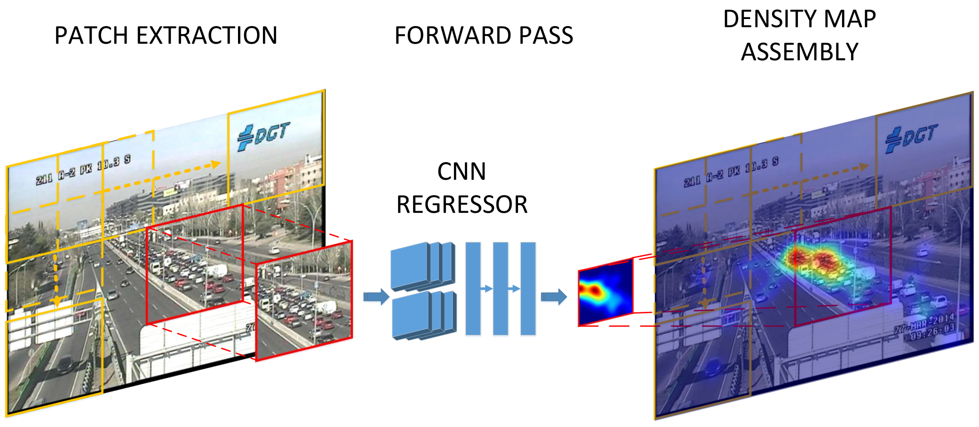


Figure 2.5: We define the object counting task like a regression problem where a deep learning model has to learn how to map image patches to object densities.

adding a novel self-gating mechanism in the skip-connections that control the information flow between the layers.

2.3 Datasets

To precisely estimate the total count of a particular object category in an image is a core task that plays a crucial role in many applications. Some direct examples are:

- Monitoring cells growing in a test tube.
- Controlling the outcome of harvest for the intelligent agriculture systems.
- Collecting statistics of the number of people attending a protest.
- Controlling the highway's congestion.

To precisely estimate the objects present in these types of scenes is not an easy task, even for a human. Building automatic counting solutions able to deal with this problem would allow the development of systems that precisely monitor those scenes, increasing the overall accuracy and releasing the human operator workload to focus on more important tasks.

To have a large and diverse dataset where we can train and test the designed models is a prerequisite for building robust and realistic machine

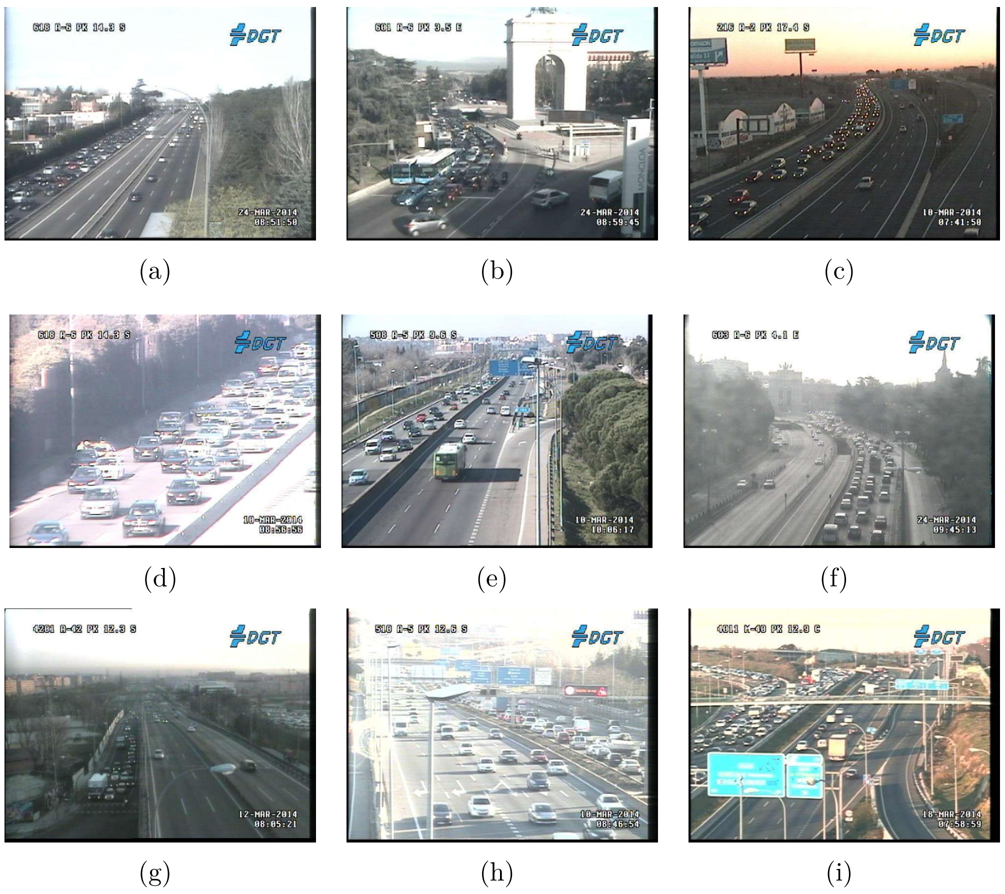


Figure 2.6: TRaffic ANd COngestionS (TRANCOS) database images. These pictures show how challenging the proposed problem of extremely overlapping vehicle counting is, even for a human.

learning models. For this reason, we run our experiments on three challenging datasets with two very different target object categories to count cars and people. In the following sections, we explain more in detail the characteristics of each dataset.

2.3.1 TRANCOS dataset

We detail in this section the TRANCOS dataset. Although there are several datasets for assessing the performance of vehicle detection approaches (*e.g.* (Everingham et al. 2010, Guerrero-Gomez-Olmedo et al. 2013, Geiger et al. 2012, Caraffi et al. 2012)), TRANCOS is the first one focused on traffic jam scenes, captured using *real* traffic surveillance cameras. Figure 2.6 shows a sample of the images provided, which illustrate how challenging the proposed problem is, *i.e.* to estimate the number of vehicles under heavy traffic condition. Note that all the provided images contain traffic congestion situations, covering a variety of different scenes and viewpoints, with changes in the light conditions and considerably different levels of overlap and crowdedness, even within the same image.

Specifically, the database consists of 1244 images. They have been acquired from a selection of public traffic surveillance cameras provided by the Directorate General of Traffic (DGT) of the Government of Spain. The cameras selected monitor different highways located in the area of Madrid, which typically presents heavy traffic congestions.

Each image has been manually annotated following a dotting annotation strategy, as in (Lempitsky and Zisserman 2010). For each image, TRANCOS provides the exact number of vehicles and their locations. In total, 46796 vehicles have been annotated. A Region of Interest (ROI) to identify the road region is also provided for each image.

The main goal of TRANCOS is to evaluate vehicle counting approaches, especially under extremely overlapping conditions. So, any method using this dataset has to predict, for each test image, the number of vehicles, and the vehicles locations. The following experimental setup has to be followed by any method using the dataset. The acquisition of the images has been done during three different weeks, which lets us distribute the images in three separate sets: training (403 images), validation (420) and test (421). Two types of training strategies are defined: 1) methods which are trained using only the provided training and validation data; 2) methods built using any data except the provided test data. In both cases, the test set must be used strictly for reporting of results alone - it must not be used in any way to train or tune systems, for example by running multiple parameter choices and reporting the best results obtained. This has to be done using the validation images, for instance.

For the evaluation metric, the novel Grid Average Mean absolute Error (GAME) is introduce by Guerrero-Gómez-Olmedo et al. (2015). The GAME metric is computed as follows,

$$GAME(L) = \frac{1}{N}\sum_{n=1}^{N}(\sum_{l=1}^{4^L}\left|D^l_{I_n} - D^l_{I_n^{gt}}\right|)\,, \tag{2.1}$$

where N is the total number of images, $D^l_{I_n}$ corresponds to the estimated object density map count for the image n and region l, and $D^l_{I_n^gt}$ is the corresponding ground truth density map for the same region. For a specific level L, GAME(L) subdivides the image using a grid of 4^L non-overlapping regions, and the error is computed as the sum of the mean absolute errors

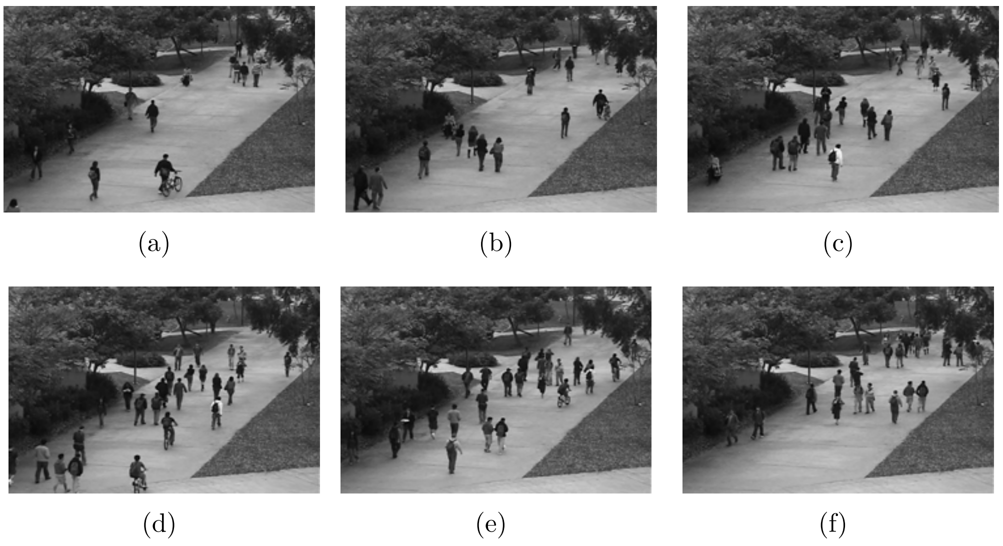


Figure 2.7: UCSD database sample images.

in each of these subregions. This metric provides a spatial measurement of the error. Note that a GAME(0) is equivalent to the Mean Absolute Error (MAE) metric:

$$MAE = \frac{1}{N} \sum_{n=1}^{N} \left| D_{I_n} - D_{I_n^{gt}} \right| . \tag{2.2}$$

With TRANCOS[1] the authors provide a set of tools for accessing the datasets and annotations described. These tools also enable the evaluation and comparison of different methods using the proposed GAME metric.

2.3.2 UCSD dataset

Here we detail the UCSD dataset (Chan et al. 2008), one of the most popular datasets of the crowd counting community. It is a 2000-frames video dataset from a surveillance camera of a single scene, recorded a 10 fps with an image resolution of 238 $\times$ 158 pixels and a single channel. Figure 2.7 shows some image samples. The images have been annotated with a dot on each pedestrian. The dataset contains a total of 49,885 pedestrian instances. It also includes a ROI and the perspective map of the scene. There are two main experimental setups. In the first experimental setup, the frames 601 to 1400 are used for the training. The remaining frames are used as test data. The second experimental setup splits the dataset into four subsets:

[1]`http://agamenon.tsc.uah.es/Personales/rlopez/data/trancos`

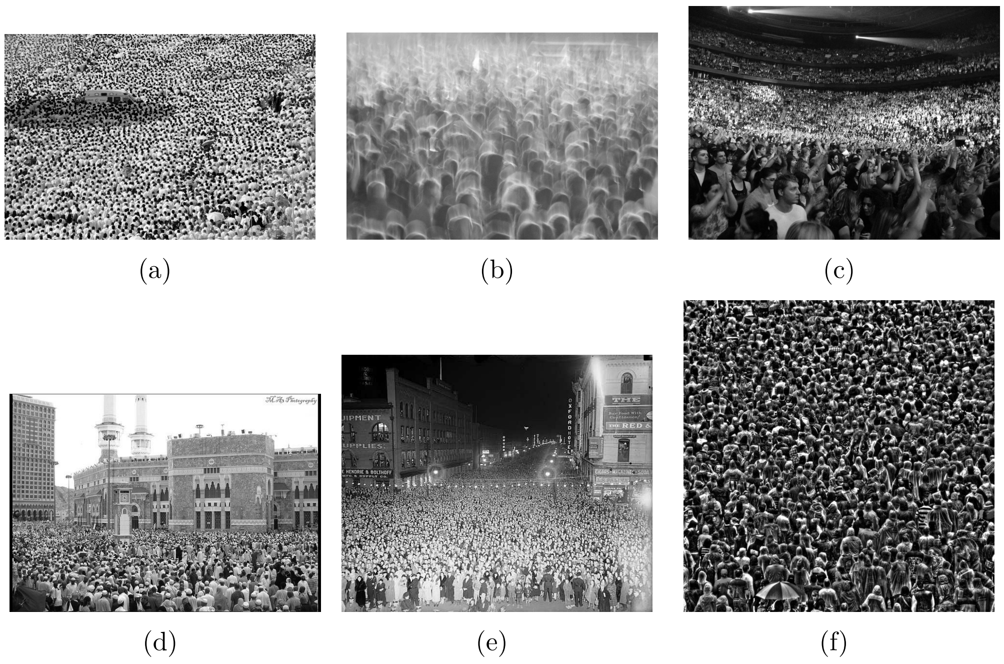


Figure 2.8: UCF_CC_50 database sample images.

- "maximal": train with frames 600:5:1400. This split is designed to analyze a model behavior when it is trained with a large number of samples.
- "downscale": train with frames 1205:5:1600. This split defines a training sequence where the number of people progressively decreases.
- "upscale": train with frames 805:5:1100. This split takes a training sequence where the number of people progressively increases.
- "minimal": train with frames 640:80:1360. This split aims to understand the model behavior when it is trained with a minimal amount of data.

All the frames out of the defined training ranges are used for testing. The evaluation metric proposed by Chan et al. (2008) is the MAE, Equation 2.2.

2.3.3 UCF_50 dataset

The **UCF_CC_50** dataset of Idrees et al. (2013) consists of 50 pictures collected from publicly available web images. The images are in grayscale of arbitrary size. The counts of persons are between 94 and 4543, with an average of 1280 individuals per image. The ground truth has been determined by manually annotating these images with dots. The images contain very crowded scenes, which belong to a diverse set of events: concerts, protests, stadiums, marathons, and pilgrimages. This dataset poses a challenging problem, especially due to the reduced number of training images and the variability between the covered scenarios. Figure 2.8 shows some image samples taken from the dataset. The dataset is usually evaluated by randomly splitting it into 5 subsets and by performing a 5-fold evaluation. In previous work, the Mean Standard Deviation (MSD) metric:

$$MSD = \frac{1}{N}\sum_{n=1}^{N} s(D_{I_n} - D_{I_n^{gt}})\,, \tag{2.3}$$

where $s(\cdot)$ is the standard derivation, and the MAE metric (see Equation 2.2) are both used for the evaluation of the models counting in this dataset.

2.3.4 ShanghayTech dataset

The ShanghaiTech dataset introduced by Zhang et al. (2016) is a publicly available and commonly used dataset for crowd counting. It contains 1198 annotated images with a total of 330,165 persons. The dataset consists of two parts: Part A contains images randomly crawled from the Internet, and Part B is made of images taken from the metropolitan areas of Shanghai. Both sets are divided into training and testing subsets, where Part A contains 300 training images and 182 images are used for testing, and Part B consists of 400 training images and 316 testing images. In Figure 2.9 we show some image samples of the dataset. The images are annotated by placing a dot on the head of each person. The dataset uses two metrics for the evaluation: the MAE of the Equation 2.2 and Mean Squared Error (MSE):

$$MSE = \sqrt{\frac{1}{N}\sum_{n=1}^{N}\left(D_{I_n} - D_{I_n^{gt}}\right)^2}. \tag{2.4}$$

2.4 Object counting notation

Let us first formalize our notation and counting objects methodology. In this work, we model the counting problem as an object density estimation task as it has been proposed by Lempitsky and Zisserman (2010).

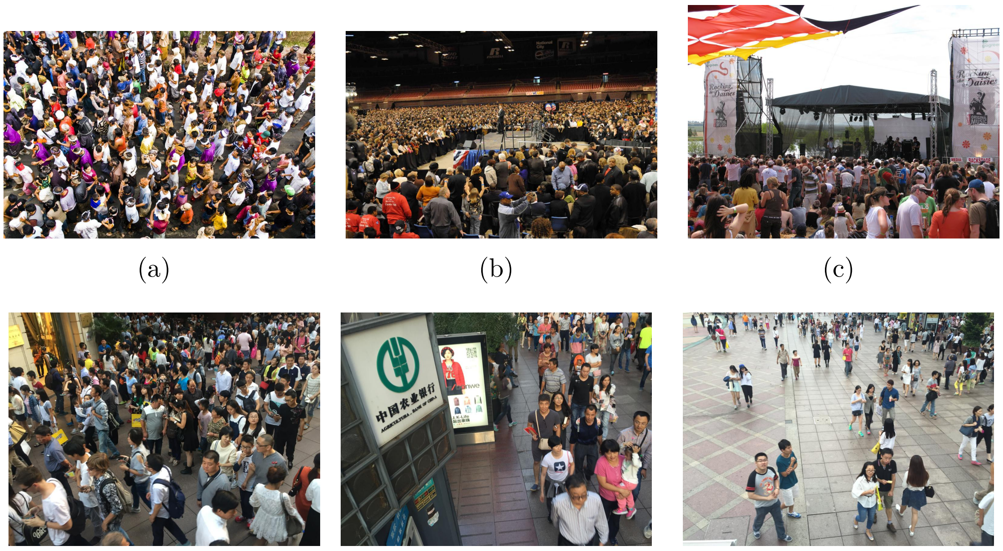


Figure 2.9: ShanghaiTech database sample images.

Our solutions require a set of annotated images, where all the objects are marked by dots. In this scenario, the ground truth density map D_I, for an image I, is defined as a sum of Gaussian functions centered on each dot annotation,

$$D_I(p) = \sum_{\mu \in \mathbf{A}_I} \mathcal{N}(p; \mu, \Sigma) \,, \tag{2.5}$$

where $\mathbf{A}_I$ is the set of 2D points annotated for the image I, and $\mathcal{N}(p; \mu, \Sigma)$ represents the evaluation of a normalized 2D Gaussian function, with mean μ and isotropic covariance matrix Σ, evaluated at pixel position defined by p. With this density map D_I, the total object count N_I can be directly obtained by integrating the density map values in D_I over the entire image, as follows,

$$N_I = \sum_{p \in I} D_I(p). \tag{2.6}$$

Note that all the Gaussian are summed, so the total object count is preserved even when there is an overlap between objects.

Given this object counting model, the main objective of our work is to design deep learning architectures able to learn the non-linear regression function $\mathcal{R}$ that takes an image patch P as an input, and returns an object density map prediction $D^{(P)}_{pred}$,

$$D^{(P)}_{pred} = \mathcal{R}(P|\Theta) \,, \tag{2.7}$$

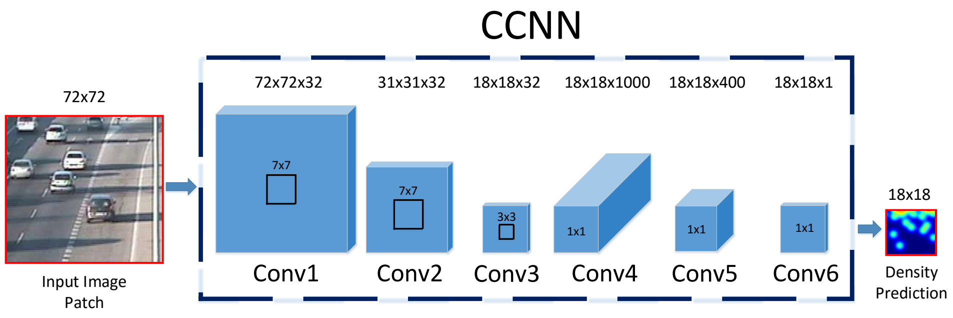


Figure 2.10: Our novel CCNN model. The input image patch is passed forward our deep network, which estimates its corresponding density map.

where Θ is the set of parameters of the CNN model. For the image patch $P \in \mathbb{R}^{h \times w \times c}$, h,w and c correspond to the height, width and number of channels of the patch, respectively. In the density prediction $D_{pred}^{(P)} \in \mathbb{R}^{h' \times w'}$, h' and w' represent the height and width of the predicted map. Thus, given an unseen test image, our model densely extracts image patches from it and generates their corresponding object density maps, which are aggregated into a density map for the whole test image.

2.5 Proposed models

In this section, we describe the designed deep neural networks architectures that have been used for the object counting. First, it is introduced the Counting CNN, a fully convolution neural network which establishes a robust baseline. Then, the Hydra architecture is detailed, a multiscale aware model for object counting. After, we introduce the Counting U-Net and the Counting GU-Net, two advanced hourglass networks that outperform the previously existing methods.

2.5.1 The Counting CNN and Hydra CNN

2.5.1.1 The Counting CNN

We introduce in this section our first deep learning architecture, the Counting CNN (CCNN). It is shown in Figure 2.10. Let us dissection it.

The CNN architecture has been implemented by using the Caffe library (Jia et al. 2014a), and it consists of 6 convolutional layers. Conv1 and Conv2 layers have filters of size 7x7 with a depth of 32, and they are followed by a max-pooling layer, with a 2x2 kernel size. The Conv3 layer has 5x5 filters with a depth of 64, and it is also followed by a max-pooling layer with another 2x2 kernel. Conv4 and Conv5 layers are made of 1x1 filters with a depth of 1000 and 400, respectively. Note that we do not integrate any fully-connected layer in the model. With these Conv4 and Conv5 layers, we propose a fully convolutional architecture (Long et al. 2014). All the previous layers are followed by rectified linear units (ReLU). Finally, Conv6 is another 1x1 filter with a depth of 1. Conv6 is in charge of returning the density map estimation $D_{pred}^{(P)}$ for the input patch P.

Like we specify in Equation (2.7), we want our deep network to learn a non-linear mapping from the appearance of an image patch to an object density map. Thus, our CCNN has to be trained to solve such a regression problem. For doing so, we connect to the Conv6 layer the following Euclidean regression loss,

$$l(\Theta) = \frac{1}{2N} \sum_{n=1}^{N} \left\| \mathcal{R}(P_n|\Theta) - D_{gt}^{(P_n)} \right\|_2^2 , \tag{2.8}$$

where N corresponds to the number of patches in the training batch, and $D_{gt}^{(P_n)}$ represents the ground-truth density for the associated training patch P_n. Recall that Θ encodes the network parameters.

How do we implement the prediction stage? Given a test image, we first densely extract image patches. As illustrated in Figure 2.10, we feed the CCNN with image patches scaled to a fixed size of 72x72 pixels. These input patches are passed through our CCNN model, which produces a density map estimation for each of them. Note that due to the two max-pooling layers, the size of the output object density map estimation is 1/4 of the size of the input image patch, *i.e.* 18x18 pixels. Therefore, all the predicted object density maps $D_{pred}^{P} = \mathcal{R}(P|\Theta)$ are rescaled in order to fit the original input patch size. Note that this rescaling generates a density map $\hat{D}_{pred}^{P}$ whose associated count does not necessarily match with the original count before the rescaling. Therefore, this new resized density map must be normalized as follows,

$$\hat{D}_{pred}^{P} = \frac{\sum_{\forall p} D_{pred}^{P}(p)}{\sum_{\forall p} \hat{D}_{pred}^{P}(p)} \hat{D}_{pred}^{P}. \tag{2.9}$$

The last step of the prediction stage consists in the assembly of all the predicted density maps for the patches. In order to generate the final object density map estimation D_{I_t}, for the given test image I_t, we simply aggregate all the predictions obtained for all the extracted patches into a unique density map of the size of the test image (see Figure 2.5). Note that due to the dense extraction of patches, the predictions will overlap, so each position of the final density map must be normalized by the number of patches that cast a prediction in it.

Like we have previously mentioned, we are not the first ones proposing a deep learning model for object counting. Zhang et al. (2015) introduce the novel Crowd CNN architecture. In a detailed comparison of both the CCNN and the Crowd CNN, we can discover the following differences. First, the network designs are different. For instance, instead of using fully-connected layers, in our CCNN we have incorporated the fully convolutional 1x1 layers Conv4, Conv5 and Conv6. This speeds up both the training a forwards pass (Long et al. 2014). Second, their learning strategy is more complex. The Crowd CNN model needs to incorporate two different loss functions (one for the density maps and one for the total count of the patches). During the optimization, they implement an iterative switching process to alternatively optimize with one loss or the other. In contrast, our CCNN *only uses one loss.* And third, our model is more compact. For the problem of crowd counting, Zhang et al. (2015) do not use the direct estimation of the Crowd CNN network to obtain the final object density estimation. Instead, they report the results feeding a ridge regressor with the output features of their Crowd CNN network. On the contrary, we do not need any extra regressor, our novel CCNN is learned in an end-to-end manner to directly predict the object density maps. Finally, our experiments reveal that the CCNN improves the results of the Crowd CNN in three of four subsets of the UCSD dataset.

2.5.1.2 The Hydra CNN

In a typical pipeline of a counting by regression model, a geometric correction of the input features, using an annotated perspective map of the scene, for instance, results fundamentally to report accurate results. This phenomenon has been described in several works, reporting state-of-the-art results (*e.g.* (Lempitsky and Zisserman 2010, Loy et al. 2013, Fiaschi et al. 2012, Zhang et al. 2015)). Technically, the perspective distortion exhibited by an image causes that features extracted from the same object but at different scene depths would have a huge difference in values. As a consequence, erroneous results are expected by models which use a *single* regression function.

With the Hydra CNN model, we want to solve this problem. That is, Hydra CNN must be a scale-aware architecture, which is not allowed to use any previous geometric correction of the scene. Our architecture should be able to learn a non-linear regression mapping, able to integrate the information

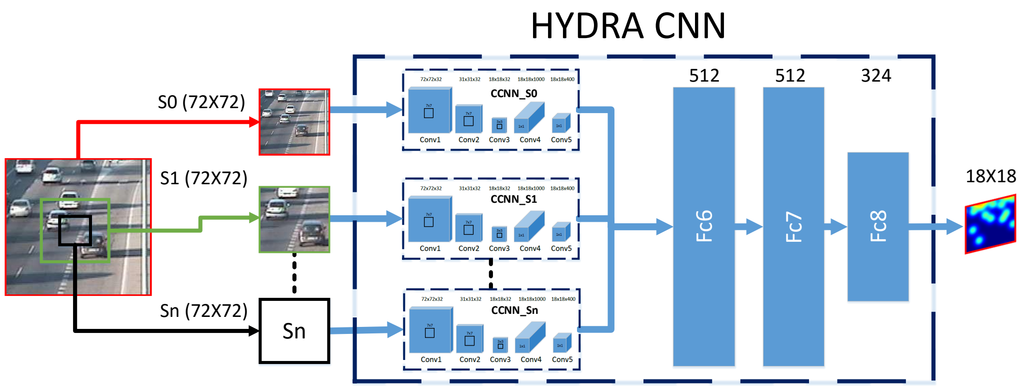


Figure 2.11: Hydra CNN. The network uses a pyramid of input patches (they are cropped and rescaled to a size of 72x72). Each level of the pyramid, representing a different scale, feeds a particular head of the Hydra. All the head outputs are concatenated and passed to a fully-connected bank of layers, which form the body of the hydra.

from multiple scales simultaneously, in order to cast a precise object density map estimation. This aspect brings a fundamental benefit: Hydra CNN can work in scenarios and datasets which consider not only a single calibrated scene. For instance, a single Hydra CNN model should be able to accurately predict the number of objects for a variety of unseen scenes, exhibiting different perspectives, and generalizing well to real-world scenarios.

We attack this problem with the idea shown in Figure 2.11. Our Hydra CNN has several *heads* and a common *body*, remembering the ancient serpentine water monster called the Hydra in Greek and Roman mythology. Each head is in charge of learning the representation for a particular scale s_i from the input pyramid of image patches. Therefore, during learning, we feed each *head* with image patches extracted at a particular scale. We have to understand the output of the heads as a set of features describing the images at different scales. Then, all these features are concatenated to feed the *body*, which is made of fully-connected layers. Notice, that the heads are not necessarily restricted to the same architecture, so their features may have different dimensions, hence the use of fully convolutional layers in the body may not be suitable. Therefore, we use a fully-connected layer in order to provide the net full access to all the head features for the different scales. Essentially, the body learns the high-dimensional representation that merges the multiscale information provided by the heads, and it is in charge of performing the final object density map estimation.

Technically, as illustrated in Figure 2.11, for each head of the Hydra CNN, we propose to use a CCNN model (CCNN_$s0$, $\ldots$, CCNN_sn). Note that we simply exclude in each CCNN model for the heads, its final Conv6 layer. Then, the outputs of the different heads are concatenated and passed to the body, where we use two fully-connected layers, with 512 neurons each one. These are the layers Fc6 and Fc7 in Figure 2.11, which are followed by a ReLu and a dropout layer. We end the architecture with the fully-connected layer Fc8, with 324 neurons, whose output is the object density map. To train this Hydra CNN model we use the same loss function defined in Equation (2.8). Again the Caffe library is used, following for the optimization the stochastic gradient descent algorithm. Finally, given a test image, we follow the same procedure described for the CCNN model to produce the final object density map estimation.

The network design of the novel Hydra CNN is inspired by the work of Li and Yu (2015) for visual saliency estimation. In (Li and Yu 2015), they propose a different network architecture but using a multiple input strategy, which combines the features of different views of the whole input image in order to return a visual saliency map. In our Hydra CNN model, we adapt this idea to use the multi-scale pyramid set of image patches to feed our network.

2.5.1.3 Experiments

We have evaluated the CCNN and the Hydra CNN solutions using three challenging benchmarks. Two have been proposed for the crowd counting problem: the UCSD pedestrian (Chan et al. 2008) and the UCF_CC_50 (Idrees et al. 2013) datasets. The third one is the TRANCOS dataset (Guerrero-Gómez-Olmedo et al. 2015), which has been designed for vehicle counting in traffic jam scenes.

TRANCOS dataset: We strictly follow the experimental setup proposed in (Guerrero-Gómez-Olmedo et al. 2015), using only the training and validation sets for learning our models. In each training image, we randomly extract 800 patches of 115x115 pixels. We also perform a data augmentation strategy by flipping each patch, having in total 1600 patches per training image. These patches are then resized to 72x72 to feed our networks. We generate the ground truth object density maps with the code provided by Guerrero-Gómez-Olmedo et al. (2015), which places a Gaussian Kernel (with a covariance matrix of $\Sigma = 15 \cdot \mathbf{1}_{2x2}$) in the center of each annotated object.

For the CCNN model, we perform a cross-validation to adjust the stan-

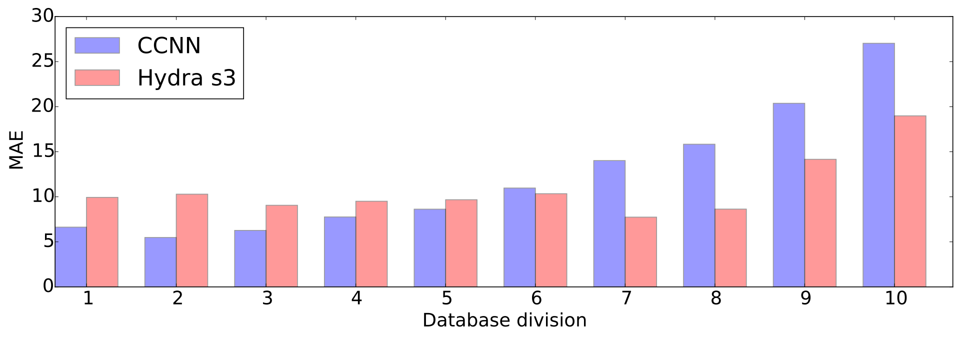


Figure 2.12: Comparison of CCNN and Hydra CNN in the TRANCOS dataset when the number of objects increases.

dard deviation values of the Gaussian noise that is necessary to initialize the weights of each layer of the deep network. The Xavier initialization method (Glorot and Bengio 2010) was used to, but with it, our CCNN models are not able to converge in our experiments.

To train the Hydra CNN, we follow the same patch extraction procedure as for the CCNN model. The only difference is that from each patch we build its corresponding pyramid of s different scales, being s the number of heads of our Hydra CNN. Therefore, the first level of the pyramid contains the original patch. For the rest of levels, we build centered and scaled crops, of size $1/s$, of the original patch. For example, in the case of a Hydra CNN with two heads, the first level of the pyramid corresponds to the original input patch, and the second level contains a crop of size 50% of the original size. When three heads are used, the second and third levels of the pyramid contain a crop of size 66% and 33% of the original size, respectively.

To initialize the heads of the Hydra CNN model, we use the same parameters discovered by the cross-validation for the CCNN. Then we perform a cross-validation to adjust the standard deviation for the layers Fc6 and Fc7.

The test is performed by densely scanning the input image with a stride of 10 pixels, and assembling all the patches as it is described in Section 2.5.1.

The TRANCOS benchmark comes with an evaluation metric to be used: the Grid Average Mean absolute Error (GAME) (Guerrero-Gómez-Olmedo et al. 2015) (see Equation 2.1). This metric provides a spatial measurement of the error. Note that a GAME(0) is equivalent to the mean absolute error (MAE) metric.

Table 2.1 shows a detailed comparison of our models with the state-of-the-art methods (Fiaschi et al. 2012), (Lempitsky and Zisserman 2010) reported in (Guerrero-Gómez-Olmedo et al. 2015).

Method	GAME 0	GAME 1	GAME 2	GAME 3
Fiaschi et al. (2012)	17.77	20.14	23.65	25.99
Lempitsky and Zisserman (2010)	13.76	16.72	20.72	24.36
CCNN (2016)	12.49	16.58	20.02	22.41
Hydra 2s (2016)	11.41	16.36	20.89	23.67
Hydra 3s (2016)	**10.99**	**13.75**	**16.69**	**19.32**
Hydra 4s (2016)	12.92	15.54	18.45	20.96
Zhang et al. (2017)	5.47	-	-	-
Li et al. (2018)	**3.56**	5.49	8.57	15.04
Laradji et al. (2018)	3.57	**4.98**	**7.42**	**11.67**

Table 2.1: TRANCOS dataset. Comparison with the of state-of-the-art models.

First, note how *all* our models outperform the previous state-of-the-art methods. The more simple architecture of CCNN already improves the results of the previously reported models by Fiaschi et al. (2012), and by Lempitsky and Zisserman (2010). Hydra CNN should be able to report the best results in TRANCOS, given the high level of variability in terms of perspective and variety of scenes that the images of this dataset exhibits. Table 2.1 shows that a Hydra CNN with just 2 scales improves the results with respect to the CCNN for a GAME(0), while for GAME(1) to GAME(3) the performance is very similar. If we go further, and train a Hydra CNN with 3 heads, we are now able to report the best results for this dataset for all the GAMES. Note how the error for the higher levels of the GAME, where this metric is more restrictive, drastically decreases. This reveals that the Hydra CNN is more precise not only predicting the object density maps, but also localizing the densities within them. If we continue increasing the number of heads of Hydra CNN, this does not guarantee an increment of the performance. On the contrary, we have experimentally observed that the model saturates for 4 heads (see the last row of Table 2.1), while the complexity dramatically increases.

Overall, these results lead us to two conclusions. First, the object density maps can be accurately and efficiently estimated using the CCNN model, which works remarkably well. Second, the Hydra CNN idea of having a pyramid of scales as input, to learn a non-linear regression model for the prediction of object density maps, seems to be more accurate, defining the novel state of the art in this benchmark.

Figure 2.12 shows an additional analysis of our models using the MAE (GAME(0)). We perform the comparison sorting all the test images by the

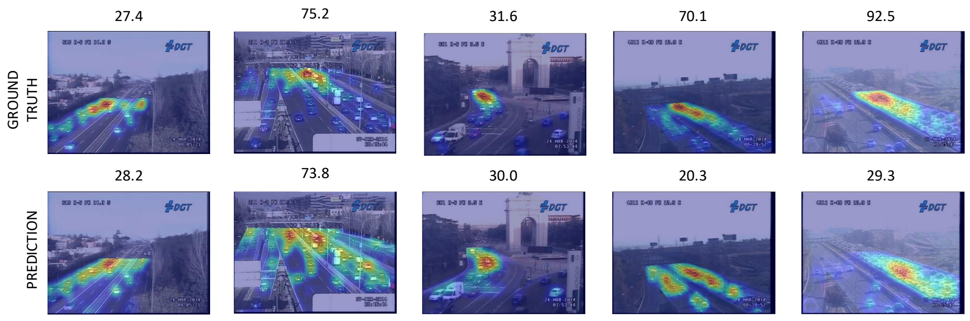


Figure 2.13: Qualitative results of our Hydra model in the TRANCOS dataset. The first row corresponds to the target image with the ground truth. The second row shows the predicted object density maps. We show the total object count above each image.

number of annotated vehicles they contain. We divide them into 10 subsets, and plot in this figure the MAE of our CCNN and Hydra CNN 3s models. Interestingly, CCNN reports a slightly lower error for the subsets of images with fewer objects. But its error quickly rises when more vehicles appear in the scene. The Hydra CNN model is clearly the winner, reporting a very stable error along the different subsets.

Finally, Figure 2.13 shows some of the qualitative results obtained. The first three images present the results where our Hydra 3s model obtains a good performance, and the last two images correspond to those for which we get the maximum error.

UCSD dataset: We follow exactly the same experimental setup that is used by Fiaschi et al. (2012), Lempitsky and Zisserman (2010), Ryan et al. (2009), and Zhang et al. (2015).

In order to train our CCNN model, for each image we collect 800 patches, of 72x72 pixels, randomly extracted all over the image and their corresponding ground truth density maps. We perform a data augmentation by flipping each patch. Therefore, in total, we have 1600 training samples per image. As usual, when the perspective map is used, the ground truth object density maps are built scaling the covariance of the 2D Gaussian kernels, where we fix a base $\Sigma = 8 \cdot \mathbf{1}_{2x2}$, as it is described by Lempitsky and Zisserman (2010).

To train the Hydra CNN models, we follow the same patch extraction detailed for the TRANCOS dataset. This time, 800 random patches of 72x72

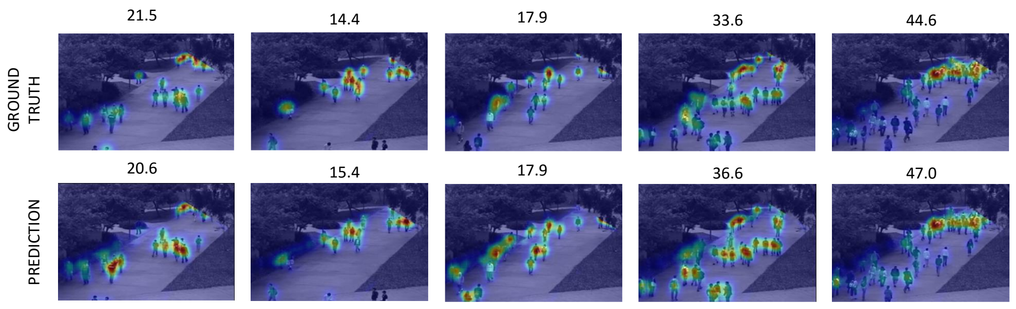


Figure 2.14: CCNN qualitative results for the UCSD dataset. The first row shows the target image with its ground truth. The second row shows the predicted object density map. We show the total object count above each image.

pixels are extracted per training image. The pyramid of scaled versions of the patches is built using the same procedure explained before. We initialize both the CCNN and the Hydra CNN models following the procedures previously explained for the TRANCOS dataset. Finally, to perform the test we fix a stride of 10 pixels and then we proceed as it is described in Section 2.5.1.

We start analyzing the performance of the CCNN model. Table 2.2 shows a comparison with all the state-of-the-art methods. Our CCNN, trained *using* the perspective map provided, like all the competing approaches, obtains the best results for the “upscale” subset. If we compare the performance of the two deep learning models, *i.e.* CCNN vs. the Crowd CNN of Zhang et al. (2015), our model gets better performance in 3 of the 4 subsets.

Method	‘maximal’	‘downscale’	‘upscale’	‘minimal’
Lempitsky and Zisserman (2010)	1.70	1.28	1.59	2.02
Fiaschi et al. (2012)	1.70	2.16	1.61	2.20
Pham et al. (2015)	1.43	1.30	1.59	1.62
Arteta et al. (2014)	**1.24**	1.31	1.69	**1.49**
Zhang et al. (2015)	1.70	**1.26**	1.59	1.52
Our CCNN	1.65	1.79	**1.11**	1.50

Table 2.2: Mean absolute error. Comparison with the state-of-the-art methods for the UCSD pedestrian dataset.

Figure 2.14 shows some qualitative results. We have chosen five frames that best represent the object density differences in the dataset. The last two frames correspond with the maximal error produced by our CCNN model.

Method	'maximal'	'downscale'	'upscale'	'minimal'
Hydra 2s	2.22	1.93	1.37	2.38
Hydra 3s	2.17	2.99	1.44	1.92

Table 2.3: MAE comparison of our Hydra 2s and Hydra 3s models trained without perspective information in the UCSD dataset.

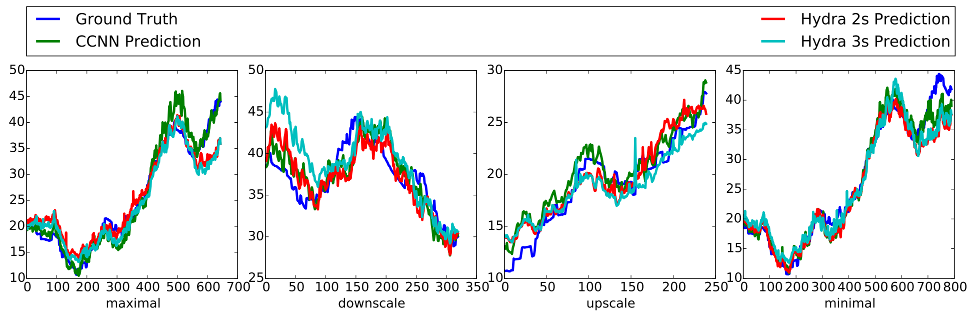


Figure 2.15: Comparison of ground truth, CCNN and Hydra CNN of two and three heads in the UCSD benchmark.

We now proceed to analyze the results obtained by the Hydra CNN models in this benchmark. Even though this dataset offers images of a fixed scene, providing its perspective map, where the objects appear at similar scales, we have decided to conduct this extra experiment with the Hydra CNN approach, to evaluate its performance with the state-of-the-art models. Table 2.3 shows the MAE results for our Hydra with two and three heads. Recall that we do not use the perspective information. We can observe two things. The first one is that both architectures report a good performance, even if they do not improve the state-of-the-art. To support this conclusion, Figure 2.15 shows a comparison between the ground truth, the CCNN model (trained using the perspective map), and the estimation generated by our Hydra with two and three heads, which does not use the perspective information. Hydra CNN models are able to closely follow both the CCNN and the GT. We believe that Hydra CNN does not outperform CCNN due to the small variability and the low perspective distortion exhibited by this dataset. In this situation, adding more scales does not seem to provide really useful information. Hence, the use of Hydra CNN does not offer here a clear advantage.

UCF_CC_50 dataset: This dataset proposes a challenging problem, especially due to the reduced number of training images, and the variability between the scenarios covered. We have followed the same experimental setup described by Idrees et al. (2013). We randomly split the dataset into 5 subsets and perform a 5-fold cross-validation. To report the results the MAE and the Mean Standard Deviation (MSD) are used.

For training our models, we scale the images in order to make the largest size equal to 800 pixels. We follow the same experimental setup described for TRANCOS. We now randomly extract 1200 image patches of 150x150 pixels with their corresponding ground truth. We also augment the training data by flipping each sample. Finally, the covariance matrix for the ground truth density map generation with the Gaussian functions is fixed to $\Sigma = 15 \cdot \mathbf{1}_{2x2}$. For the initialization of the CCNN and the Hydra CNN models, we follow the cross-validation procedure already described for the other datasets. To do the test, we densely scan the image with a stride of 10 pixels.

Table 2.4 shows a comparison of our models with the state-of-the-art approaches. At the moment of the release of this work, the best performance is given by our Hydra CNN 2s, which is able to drastically reduce the MAE. Hydra CNN with 3 scales outperforms 3 of 5 models previously published. The CCNN approach only improves the results reported in (Rodriguez et al. 2011), and (Lempitsky and Zisserman 2010). Analyzing the results, we find that the performance of the CCNN decreases especially in those images with the highest number of humans and where the perspective really matters. This issue and the results provided, confirm the advantages of the scale-aware Hydra model for the very crowded scenes of the UCF_CC_50 dataset. Currently, the state of the art is defined by the posterior work of Cao et al. (2018), which is closely followed by Ranjan et al. (2018).

Method	MAE	MSD
Rodriguez et al. (2011)	655.7	697.8
Lempitsky and Zisserman (2010)	493.4	487.1
Zhang et al. (2015)	467.0	498.5
Idrees et al. (2013)	419.5	541.6
Zhang et al. (2016)	377.6	509.1
Ranjan et al. (2018)	260.9	365.5
Cao et al. (2018)	**258.4**	**334.0**
CCNN	488.67	646.68
Hydra 2s	333.73	425.26
Hydra 3s	465.73	371.84

Table 2.4: MAE and MSD comparison for the UCF_CC_50 dataset.

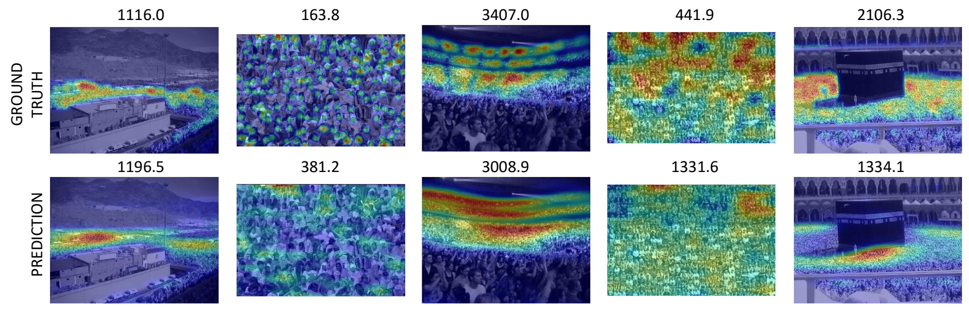


Figure 2.16: UCF_CC_50 dataset qualitative results for Hydra CNN with two scales. First row corresponds to the target image with the GT. Second row shows the predicted object density maps. We show the total object count above each image.

Figure 2.16 shows some of the qualitative results that are obtained by our Hydra CNN model with two heads. The first three columns correspond with results where our network reports a good performance, while the last two columns show the maximum errors.

2.5.2 The counting U-Net and GU-Net architectures

2.5.2.1 The counting U-Net

U-Net is an hourglass-like CNN that has been recently proposed by Ronneberger et al. (2015) for segmentation. U-Nets are fully convolutional neural networks which can be divided into a compression and a construction part. Both parts are connected with short-cut connections leading from a compression layer to a construction layer with the same feature size. This type of architecture has shown excellent performance on mapping images into labels maps of the same size of the input. Figure 2.17 shows the proposed U-Net architecture for the counting task. The model consists of 11 layers. The first five layers are convolutional layers that address the compression part of a U-Net. The following five layers are transpose-convolutional layers (Long et al. 2015) that perform the construction. Finally, the eleventh layer is the output layer. The first convolutional layer has 64 filters of size $[4, 4]$, the second convolutional layer has 128 filters of size $[4, 4]$, and the third, fourth and fifth layers have 128 filters of size $[3, 3]$. In all the convolutional layers we use a stride of $[2, 2]$ to reduce the size of the features maps by one half. The remaining layers use the transpose-convolution introduced by previous

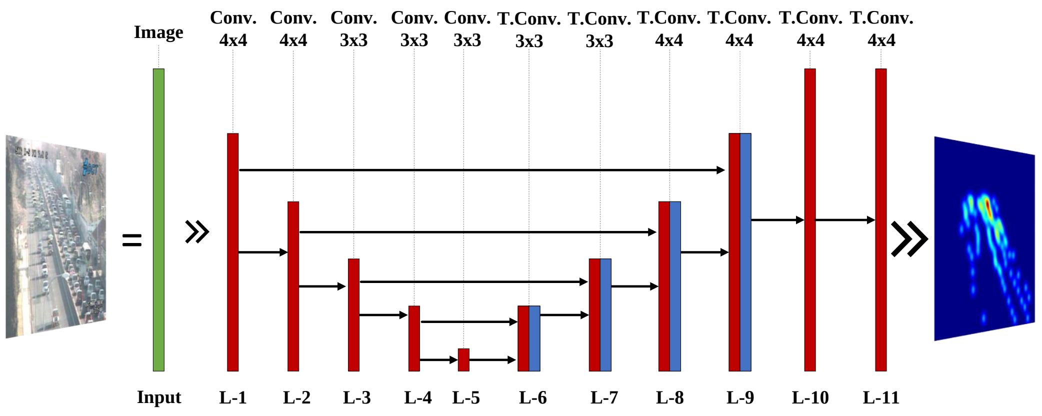


Figure 2.17: U-Net Network architecture for object counting.

work of Long et al. (2015). The sixth and seventh layers have 128 filters with a size of $[3, 3]$, the eighth and ninth layers have 128 filters of $[4, 4]$. The tenth has 64 filters with a size of $[4, 4]$ in order to match the first convolutional layers. In all of the expansion layers, we use a stride of $[2, 2]$ which doubles the size of the feature maps. As the activation function, we make use of the Leaky ReLu introduced in (Maas et al. 2013). For the eleventh layer, we use a transpose-convolution with a single filter, the size of $[4, 4]$, and a stride of $[1, 1]$. This operation does not perform any *up-sampling*, it just produces an output of the model. Also, notice that for this layer the usage of the convolution or the transpose-convolution is equivalent. Due to this is the output layer, there is no activation function needed. As in the original U-Net architecture, the *short-cuts* are connecting the compression and expansion layers with the same feature map size, *i.e.* the first layer connects with the ninth, the second with the eight, the third with the seventh and the fourth with the sixth. For the fusion operation, we chose the concatenation as in (Ronneberger et al. 2015), although summation, point-wise multiplication or other operation might also be suitable.

The proposed U-Net architecture has a receptive field of a region of $[96, 96]$ pixels. Since we test our algorithm for the counting problem, we have setup the presented receptive field to cover an area that can contain an object of our datasets, surrounded by background. The size of the filter is chosen to be $[4, 4]$ for those features maps that are greater or equal to $[24, 24]$, while for those that are smaller we used a slightly reduced size of $[3, 3]$. It is important to notice that the bottleneck occurs in the fifth layer. It receives a feature map with a size of $[3, 3]$ as input and, therefore, with the filter size of that layer, the receptive field of the network covers the entire input image.

In contrast with the originally proposed U-Net, the presented network

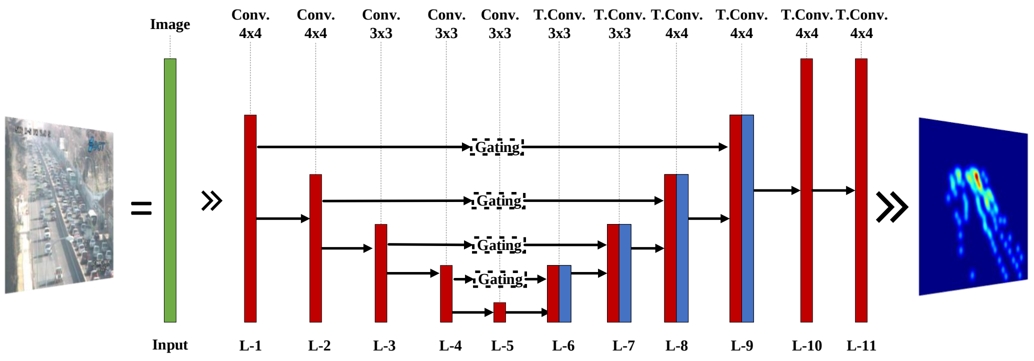


Figure 2.18: Gated U-Net Network architecture.

architecture is focused on solving the object density estimation in images, and similarly to the previous models, it can be fitted by minimizing the loss function of Equation 2.8.

2.5.2.2 The counting GU-Net

In this section, we introduce GU-Net, an extension of the previously proposed counting U-Net that results from the integration of learnable short-cut units. Figure 2.18 illustrates the proposed model architecture. We add four short-cut connection units, one between each pair of layers for which there is a short-cut in the U-Net. The first learnable short-cut connection $G1$ consists of a convolutional layer with 64 filters each with a size of $[4, 4]$. The second learnable short-cut connection $G2$ is made up of 128 filters each of size $[4, 4]$. The third learnable short-cut unit $G3$ consists of a convolutional layer of 128 filters of size $[3, 3]$. The fourth unit $G4$ has 128 filters of size $[3, 3]$. All convolutional layers of the short-cut units work with a stride of $[1, 1]$.

What are these learnable short-cut connection for? Short-cut connections are connections between layers in deep neural networks which skip at least one intermediate layer. These connections have proven to be beneficial in deep neural networks applied to problems such as object recognition (He et al. 2016). Short-cut connections enable the training of deeper networks with backpropagation. A more recent and alternative explanation is that networks with short-cut connections behave similarly to ensembles of shallower networks (Veit et al. 2016). In all existing state-of-the-art neural networks, short-cut connections either exist or do not exist between layers. We propose a framework for fine-grained learning of the strength of short-cut connections between layers.

In the presented GU-Net model for object counting, the gated short-cut

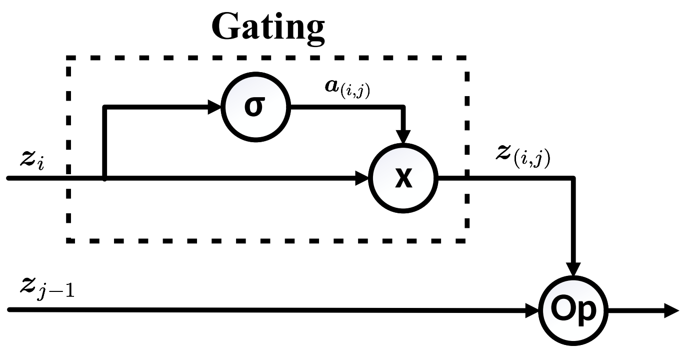


Figure 2.19: Gating operation of the short-cuts.

units act as a bit-wise soft mask that determines the amount of information to pass to the respective layers so as to improve the quality of the feature maps for counting. In a deep neural network, the first layers are specialized in detecting low-level features such as edges, textures, colors, etc. These low-level features are needed in a normal feed-forward neural network, but when they are combined with deeper features, they might or might not contribute to the overall performance as the deeper features and to add noise to the learning process. For this reason, the gating units are especially useful to automate the feature selection mechanism for improving the *short-cuts* connections while strongly back propagating the updates to the early layers.

The main limitation of the presented idea is that the gating units do not add more freedom to the general network but add additional parameters (one additional convolutional layer per short-cut connection). Therefore, the fitting capability of GU-Net is the same as for the original U-Net. However, the gating strategy leads to more robust models that produce better results.

With respect to the training procedure, the counting GU-Net does not require any special modification with respect to the previously introduced counting U-Net and it is trained to minimize the loss function of Equation 2.8.

2.5.2.3 Learning short-cut connections for object counting

Technically, we propose a gating mechanism that can be applied to any deep neural network to learn short-cuts and to optimize the flow of information between earlier and later layers of a network architecture. Let us first, however, introduce some background terminology.

Taking the definition of the Equation 2.7, a neural network $R(P|\Theta)$ can be written as a composition of functions $R(P|\Theta) = f^n_{\Theta_n} \circ \dots f^2_{\Theta_2} \circ f^1_{\Theta_1}(P)$, where each f^i represents the transformation function for layer $i \in \{1, ..., n\}$ with its parameters Θ_i. Each layer maps its input to an output tensor $\boldsymbol{z}_i \in \mathbb{R}^{[h',w',d']}$.

Typically, the output $\boldsymbol{z}_i$ of layer i is used as input to the proceeding layer $f^{i+1}_{\Theta_{i+1}}(\boldsymbol{z}_i)$ to generate output $\boldsymbol{z}_{i+1}$, and so on. Each layer, therefore, is only connected locally to its preceding and proceeding layers.

However, we can also connect the output of a layer i to the input of a layer j with $j > i + 1$, that is, we can create *short-cut connections* that skip a number of layers in the network. The first class of network architectures popularizing these skip-connections were introduced with the ResNet model class (He et al. 2016). Here, we do not only create certain short-cut connections but learn their connection strength using a *gated short-cut unit* (see Figure 2.19). The gated short-cut units determine the amount of information which is passed to other layers and also the ways in which this information is combined with the input of these later layers.

Whenever a layer i with transformation function f^i is connected to a layer j with $j > i + 1$ we introduce a new convolutional layer $g^{(i,j)}$ whose hyperparameters are identical to that of layer i. With $\boldsymbol{z}_i$ being the output of layer i we compute

$$\boldsymbol{a}_{(i,j)} = \sigma(g^{(i,j)}(\boldsymbol{z}_i)), \tag{2.10}$$

where σ is the sigmoid function. The tensor $\boldsymbol{a}_{(i,j)}$ consists of scalars between 0 and 1 that determine the amount of information passed through to layer j for each of the local convolutions applied to $\boldsymbol{z}_i$. Hence, we then compute the element-wise product of the output of layer i and the tensor $\boldsymbol{a}_{(i,j)}$

$$\boldsymbol{z}_{(i,j)} = \boldsymbol{z}_i \odot \boldsymbol{a}_{(i,j)}, \tag{2.11}$$

where $\boldsymbol{a}_{(i,j)}$ are again the values of the gating function that decides how much information is let through.

Figure 2.19 illustrates the gating unit. $\boldsymbol{z}_i$ is used as input to a convolutinal layer to compute the gating scores $\boldsymbol{a}_{(i,j)}$. A point-wise multiplication is performed to obtain $\boldsymbol{z}_{(i,j)}$. Those gated features are then combined with the input $\boldsymbol{z}_{j-1}$ of layer j by combining $\boldsymbol{z}_{(i,j)}$ and $\boldsymbol{z}_{j-1}$ using an operation $\circ$ such as the sum, point-wise multiplication, or concatenation. The resulting tensor $\boldsymbol{z}_{(i,j)} \circ \boldsymbol{z}_{j-1}$ is then used as the input for layer j.

2.5.2.4 Experiments

We have conducted experiments on three publicly available object counting datasets: TRANCOS (Guerrero-Gómez-Olmedo et al. 2015), ShanghaiTech (Zhang et al. 2016), and UCSD (Chan et al. 2008). We perform a detailed comparison with state-of-the-art object counting methods. Moreover, we provide empirical insights into the advantages of the learnable short-cut connections introduced.

Model	GAME 0	GAME 1	GAME 2	GAME 3
Ew. Multiplication	6.19	7.24	8.64	10.51
Summation	4.81	6.09	7.63	9.60
Concatenation	**4.44**	**5.84**	**7.34**	**9.29**

Table 2.5: Fusion operation comparison in the TRANCOS dataset and the GU-Net model.

We use Tensorflow library (Abadi et al. 2015) to implement the proposed U-Net and GU-Net models for object counting. To train our models, we initialize all the weights with samples from a normal distribution with mean zero and standard deviation of 0.02. We make use of the $L2$ regularizer with a scale value of 2.5×10^{-5} in all layers. To perform gradient descent we use Adam (Kingma and Ba 2015) with a learning rate of 10^{-4}, β_1 of 0.9, and β_2 of 0.999. These are hyperparameter values commonly used with this optimizer. We train our systems for 2×10^5 iterations, with mini-batches consisting of 128 patches. The patches of each mini-batch are extracted from a random image of the training set such that 50% of the patches contain a centered object, and the remaining patches are randomly sampled from the image. We perform data augmentation by flipping images horizontally with a probability of 0.5. All the pixel values from all channels are scaled to the range $[0, 1]$. The ground truth of each dataset is generated by placing a normal distribution on top of each of the annotated objects in the image. The standard deviation σ of the normal distribution varies depending on the dataset under consideration.

To perform the object count we feed entire images to our models. Notice that the proposed models are fully convolutional neural networks, therefore they are not constrained to a fixed template-size. Hence, we can work with input images of different sizes and aspect ratios, and the output of the network is a density map that matches the input size. Notice that our previous models produce density maps that are significantly smaller than the input, and the Hydra architecture is constrained to a fixed template-size which makes the model less flexible and significantly slower. Moreover, the presented models have significantly fewer parameters than most of the state-of-the-art methods. To analyze an image of size $[640, 480, 3]$ with a GU-Net architecture takes on average only 23ms when executed in a conventional NVIDIA GeForce 1080 Ti.

We use Mean Absolute Error (MAE) see Equation 2.2 and Mean Squared Error (MSE), see Equation 2.3 for the evaluation of the results on ShanghaiTech, and UCSD. The GAME metric (Equation 2.1) is used for the evaluation of the TRANCOS dataset.

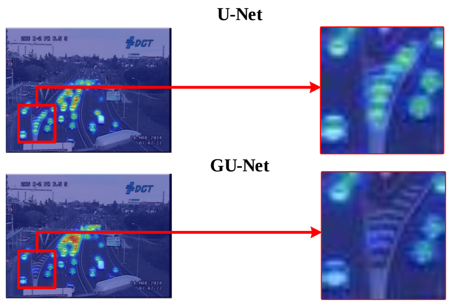
Trancos				
Model	GAME 0	GAME 1	GAME 2	GAME 3
Fiaschi et al. (2012)	17.77	20.14	23.65	25.99
Lempitsky and Zisserman (2010)	13.76	16.72	20.72	24.36
Hydra 3s (2016)	10.99	13.75	16.69	19.32
Zhang et al. (2017)	5.47	-	-	-
Li et al. (2018)	**3.56**	5.49	8.57	15.04
Laradji et al. (2018)	3.57	**4.98**	7.42	11.67
U-Net (2018)	4.58	6.69	8.69	10.83
GU-Net (2018)	4.44	5.84	**7.34**	**9.29**

Table 2.6: Results comparison for TRANCOS dataset.

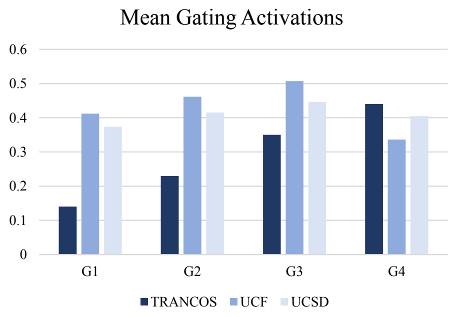
TRANCOS experiment: to train all of our experiment on this dataset, we generate the ground truth density maps by setting the standard deviation of the normal distributions to $\sigma = 10$. In addition to the general setup, we also perform random gamma transformation to the images.

Since we are trying to fuse features from low a high-level layer with *short-cuts*, an important aspect to investigate is the operation $\circ$, which is the one used for the feature combination. To this end, we perform experiments with the element-wise multiplication and summation operations as well as the concatenation operation. Table 2.5 lists the results for the GU-Net and each of the operations described. We observe that the element-wise multiplication, even though it produces accurate density maps, offers the lowest performance. The summation is a linear operation that shows satisfying results. It also has the advantage that it does not add channels to the resulting features, making this type of operation especially suitable for situations in which memory consumption and processing capabilities are constrained. The best performing operation is the concatenation. Therefore, for the remainder of our experiments, we use the concatenation operation as the merging operation of the short-cut connections.

Table 2.6 details the results of the various methods on the TRANCOS dataset. Shallow methods are listed in the upper part of the table. In the second part are the deep learning methods. The last two rows list the results obtained by the U-Net and the GU-Net (U-Net + learnable short-cuts) models. The GU-Net achieves better results compared to the U-Net base architecture. These improvements are consistent across the various GAME settings we use. As an overall comparison, we appreciate how the swallow methods are outperformed by deep learning based approaches. Among the deep learning approaches, the very recent work of Li et al. (2018) gets the



(a)



(b)

Figure 2.20: a) Qualitative error analysis between U-Net and GU-Net. (b) Mean activation scores of the gating units of the analyzed datasets.

best performance in special for GAME(0) (or MAE), but it reports a high error for GAME(3). This is probably indicating that the model is under-counting and over-counting in different regions of the image, but the overall count is accurate due to the error in one image area is compensated by the error of another area. The best performance for GAME(2) and GAME(3) is given by our GU-Net. It shows the robustness of the proposed model. Figure 2.21 shows, a few qualitative results of the proposed models.

Adding learnable short-cut units helps to determine the strength of the information that is passed to the later construction layers. It increases the robustness of the final model by blocking information that is not relevant for these layers. Intuitively, in a deep convolutional neural network, the first layers learn to detect low-level features such as edges, textures, or colors. The later layers are able to capture more complex feature patterns such as eyes, legs, windows, etc. Therefore, when low-level features are combined with high-level features, the resulting feature map might contain irrelevant information potentially adding noise. As an example, in Figure 2.20(a) we show a real example of a situation that clearly shows how a low-level texture is adding noise. The U-Net is confusing the road lines as cars while the GU-Net could efficiently handle the problem. The learnable short-cut units of GU-Net learn to detect these situations and effectively block the forwarded information of short-cut connections. To explore this hypothesis more thoroughly, we measure the mean values of the activation scalars (the output of the sigmoids) for the learnable short-cut connections in the GU-Net for several datasets. Figure 2.20(b) depicts the results of this experiment. It shows the effect of the short-cut units is data dependent, that is, the learnable short-cut units automatically adapt to the dataset under consideration.

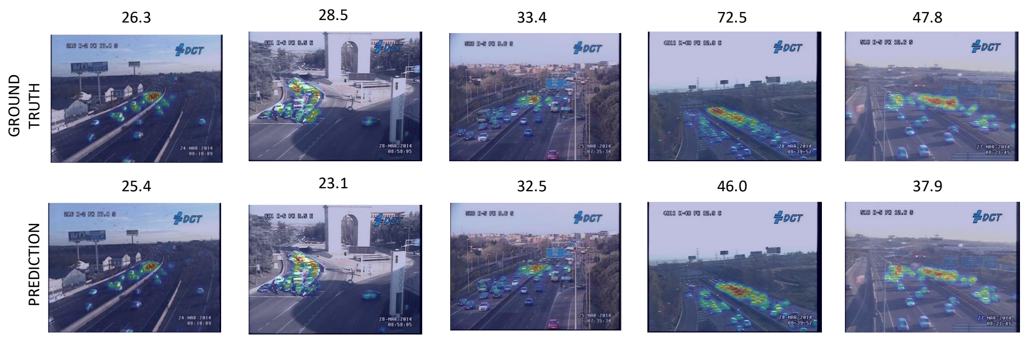


Figure 2.21: Trancos qualitative results.

ShanghaiTech experiment: The dataset consists of two parts: Part A contains images randomly crawled on the Internet, and Part B is made of images taken from the metropolitan areas of Shanghai. Both sets are divided into training and testing, where Part A contains 300 training images and 182 images are used for testing, and Part B consists of 400 training images and 316 testing images. In addition to the standard sets, we created our own validation set for each part by randomly taking 50 images out of the training sets which are used to tune our parameters. We resize the images to have a maximum height or width of 380 pixels, and we generate the ground truth density maps by placing a Gaussian kernel on each annotated position with a standard derivation of $\sigma = 4$. We follow the training procedure described at the beginning of Section 2.5.2.4. Table 2.7 lists the results. Consistent with the previous experiments, the proposed gating mechanism improves the performance of our U-Net baseline model. The current state of the art is now defined in both parts of the dataset by some posterior works. Currently, the best performance is achieved by Cao et al. (2018), and it is closely followed by Li et al. (2018) and Ranjan et al. (2018). The proposed models achieve a good performance in both datasets. Especially in Part B, our models obtain the second best results from all the previous works, just closely overtaken by Li et al. (2018). Moreover, notice that all the best-performing methods are significantly larger models. They range from 7.9M of learnable parameters of the model proposed by Ranjan et al. (2018), and they go up to 63M of the model proposed by Sindagi and Patel (2017), while our most complex model has only 2.8M parameters. In Figure 2.22 we show some qualitative results of our GU-Net model.

UCSD experiment: The UCSD dataset is a standard benchmark in the crowd counting community. As in previous work of Chan et al. (2008), we

ShanghaiTech				
Model	Part A		Part B	
	MAE	MSE	MAE	MSE
Zhang et al. (2015)	181.8	277.7	32.0	49.8
Zhang et al. (2016)	110.2	173.2	26.4	41.3
Sindagi and Patel (2017)	73.6	106.4	20.1	30.1
Laradji et al. (2018)	-	-	21.6	-
Ranjan et al. (2018)	68.5	116.2	10.7	16.0
Li et al. (2018)	68.2	115.0	10.6	16.0
Cao et al. (2018)	**67.0**	**104.5**	**8.4**	**13.6**
U-Net (2018)	104.9	173.3	17.1	25.8
GU-Net (2018)	101.4	167.8	14.7	23.3

Table 2.7: ShanghaiTech state-of-the-art.

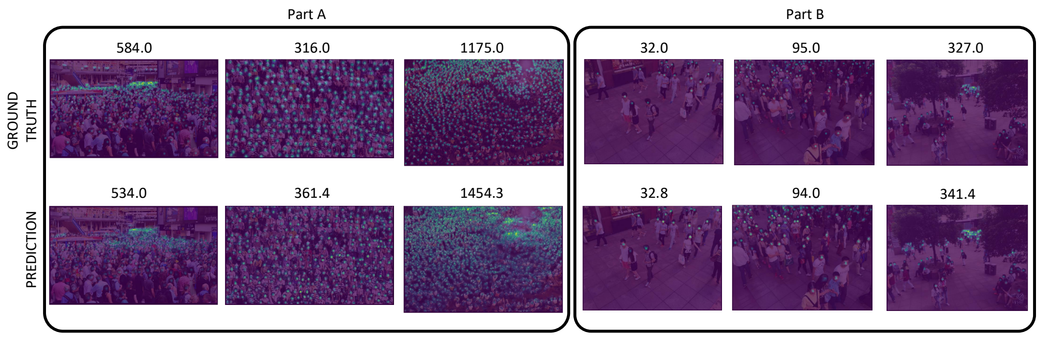


Figure 2.22: GU-Net qualitative results for ShanghaiTech.

train the models on frames 601 to 1400. The remaining frames are used as test data. For our experiments, we sample 100 frames uniformly at random from the set of training frames and use them as our validation set. To generate the ground truth density maps, we set the standard deviation of the normal distributions placed on the objects to $\sigma = 5$. To train our models, we follow the procedure described at the beginning of Section 2.5.2.4. Table 2.8 lists the results for the UCSD dataset. We reach state-of-the-art results, but also we observe how GU-Net model consistently improves our U-Net. Our solutions are outperformed by the work of Li et al. (2018), Zhang et al. (2016), Cao et al. (2018), and Laradji et al. (2018) models. However, we should notice that our models are not pretrained. Moreover, it is worth to mention that our architectures have much less learnable parameter than these previous works, and with them we are able to recover much of their capacity, reporting close

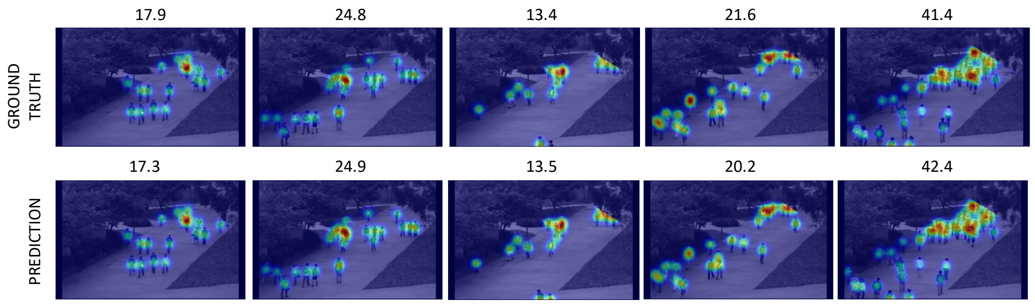


Figure 2.23: GU-Net qualitative results for UCSD.

MAEs. In the Figure 2.23 we show some qualitative results obtained by the proposed GU-Net model.

UCSD		
Model	MAE	MSE
Zhang et al. (2017)	1.67	3.41
Pham et al. (2015)	1.61	4.4
Zhang et al. (2015)	1.6	5.97
Li et al. (2018)	1.16	1.47
Zhang et al. (2016)	1.07	1.35
Cao et al. (2018)	1.02	**1.29**
Laradji et al. (2018)	**1.01**	-
U-Net	1.28	1.57
GU-Net	1.25	1.59

Table 2.8: UCSD state-of-the-art.

2.6 Discussion

This chapter has detailed four models that were published in two different top-tier conferences with a gap of 2 years in between of them. CCNN and Hydra were presented in (Oñoro-Rubio and López-Sastre 2016) during the early days of object counting with deep learning, while the counting U-Net and GU-Net were recently presented in (Oñoro-Rubio et al. 2018). These last models take advantage of the latest advances in the deep learning field and considerably improve our previous approaches.

The object scale introduced by the perspective of the scene is the major problem that has been addressed during the first publication (Oñoro-Rubio

Trancos				
Model	GAME 0	GAME 1	GAME 2	GAME 3
CCNN (2016)	12.49	16.58	20.02	22.41
Hydra 2s (2016)	11.41	16.36	20.89	23.67
Hydra 3s (2016)	10.99	13.75	16.69	19.32
U-Net (2018)	4.58	6.69	8.69	10.83
GU-Net (2018)	**4.44**	**5.84**	**7.34**	**9.29**

Table 2.9: Results comparison of the proposed models in TRANCOS dataset.

and López-Sastre 2016). The CCNN presented in this work is, to be the best of our knowledge, the first fully convolutional neural network solution for object counting. CCNN acts as a strong baseline for the perspective aware solution introduced by Hydra. Having a multiscale model is an idea that was almost simultaneously explored by other teams, *e.g.* (Zhang et al. 2016). In their model, for instance, they proposed a single input and a multiple column CNN which progressively increases the filter size in order to have a multiscale model. However, since all the columns are trained with the same image scale, all the columns learn to map the same image scale distribution, and the filter size would only impact in the receptive field of the column branch and the number of parameters. Later on, Liu et al. (2017) proposed a very similar idea to our Hydra model for metastases detection.

Having multiple scale views of the input is an efficient way to increase the information fed into the model. However, the Hydra model has limitations. The first limitation is given by the need of a fixed template size. Since a fully-connected layer merges the information given by the heads, the input to each head needs to be processed in order to fit a fixed template size. This design significantly increases the processing time and slows down the forward pass of the network. As for the second limitation, the output size produced by the network is significantly smaller than the corresponding ground truth.

The introduced counting U-Net solution already addresses the main limitations of the Hydra model. The proposed design eliminates the need of the fully-connected layers, making the model template size free: the integration of the transpose-convolution in the decoder allows the network to produce outputs with the same size of the inputs. Finally, the short-cut connections that forward information with multiple receptive fields allow the network to naturally handle multiple scales. The proposed self-gated mechanism of the GU-Net model acts as a feature selection mechanism over the short-cut connection that improves the U-Net baseline by blocking noisy information over the short-cut.

GU-Net entails an improvement over the U-Net baseline. The improvement is especially prominent in datasets with scenes where the objects are extremely overlapped, and where there is a large variance on the objects sizes. Two examples are TRANCOS (see Table 2.6) or ShanghaiTech (see Table 2.7). On the other hand, in situations where the overlap and the objects sizes are not so remarkable (*e.g.* UCSD), the proposed self-gating mechanism does not necessarily bring an improvement, but it adds an overhead to the model in terms of parameters and computation time. By taking a look at Table 2.8, it can be observed a small improvement of the GU-Net with respect to the U-Net baseline in terms of MAE, but a small loose in terms of MSE.

Table 2.9 joins the results of all the proposed models using the TRANCOS dataset. The first half of the table groups the CCNN and the Hydra models proposed in 2016. The second half joins the results produced by the counting U-Net and GU-Net presented in 2018. It can be clearly observed that the GAME(3) error has been reduced significantly by the U-Net and the GU-Net solutions.

2.7 Conclusions

In this chapter have presented a complete overview of the state of the art in object counting. We have also introduced four new approaches for the object counting problem.

With our first architecture, the CCNN model, we show that object density maps can be accurately and efficiently estimated, letting the network to learn the direct mapping which transforms the appearance of image patches into object density maps. We are able to match and improve the counting accuracy of much more complex models, such as (Zhang et al. 2015), where multiple loss functions and extra regressors are used in conjunction with the deep model.

Our second model, Hydra CNN, goes one step further and provides a scale-aware solution, which is designed to learn a non-linear regressor to generate the object density maps from a pyramid of image patches at multiple scales. The experimental validation reveals that Hydra not only improves the results of its predecessor, our CCNN, but also that it is able to improve the state of the art for those benchmarks that propose to count objects in different scenes, showing very crowded situations, and where no geometric information for the scene is provided (*e.g.* its perspective map).

By making our software and pre-trained models available[2], we make it ef-

[2]https://github.com/gramuah/ccnn

fortless for future researches to reproduce our results and to facilitate further progress towards more accurate solutions for this challenging task.

In addition to the previous contributions, we have created two more deep learning architectures for the object counting problem: U-Net and GU-Net. We design our U-Net like architecture as a baseline. For the GU-Net architecture, we introduce the novel learnable gatings idea. The proposed gating mechanism improves the existing baseline by making it robust to certain types of noisy situations. We have demonstrated and discussed the advantages and limitations of the proposed approach by conducting experiments on three standard benchmark datasets in the object counting domain.

Finally, it should be noted that all the proposed counting solutions have been evaluated specifically on the problem of vehicle counting, thanks to the TRANCOS database. The results are promising, as the implemented solutions are able to monitor these scenes with great precision. This will make it possible to implement real applications that allow users, knowing the degree of congestion of the roads that are on their route, to automatically search for alternative routes, or even to make predictions of which route would be better to take tomorrow, given the history of congestion evaluated by the system.

Chapter 3

Simultaneous Object Localization and Pose Estimation

Our intelligence is what makes us human, and AI is an extension of that quality.

Yann LeCun.

3.1 Introduction

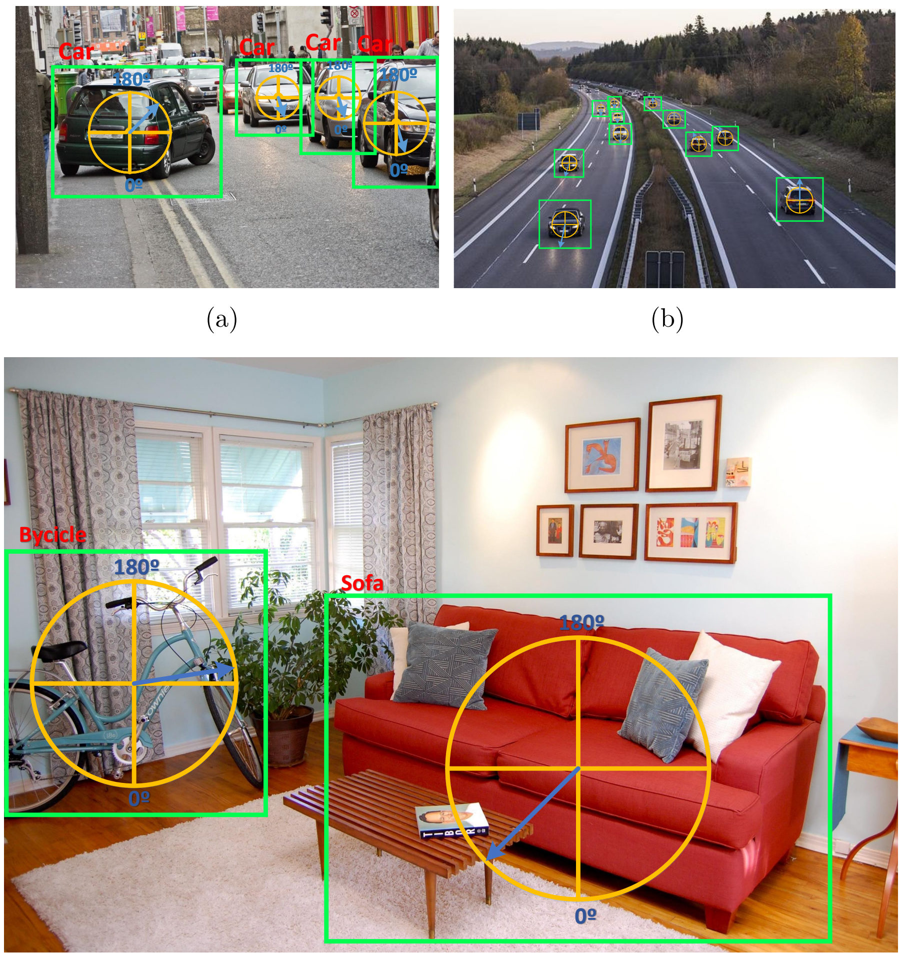
Detecting objects and estimating their pose remains as one of the major challenges of the computer vision research community. There exists a compromise between localizing the objects and estimating their viewpoints. The detector ideally needs to be view-invariant, while the pose estimation process should be able to generalize towards the category-level. This chapter is an exploration of using deep learning models for solving both problems simultaneously. For doing so, we propose three novel deep learning architectures, which are able to perform a joint detection and pose estimation, where we gradually decouple the two tasks. We also investigate whether the pose estimation problem should be solved as a classification or regression problem, is this still an open question in the computer vision community. We detail a comparative analysis of all our solutions and the methods that currently define the state of the art for this problem. We use PASCAL3D+ (Xiang et al. 2014) and ObjectNet3D (Xiang et al. 2016) datasets to present the thorough experimental evaluation and main results. With the proposed models we achieve the state of the art performance in both datasets.

Over the last decades, the category-level object detection problem has drawn considerable attention. As a result, much progress has been realized, led mainly by international challenges and benchmarking datasets, such as the PASCAL VOC Challenges (Everingham et al. 2010) or the ImageNet dataset (Deng et al. 2009). Nevertheless, researchers soon identified the importance of not only localizing the objects, but also estimating their poses or viewpoints, *e.g.* (Thomas et al. 2006), (Savarese 2008), (Lopez-Sastre et al. 2011), and (Yingze-Bao et al. 2011). This new capability results fundamentally to enable a true interaction with the world and its objects. For instance, a robot which merely knows the location of a cup but that cannot find its handle, will not be able to grasp it. In the end, the robotic solution needs to know a viewpoint estimation of the object to facilitate the inference of the visual affordance for the object. Also, in the augmented reality field, to localize and estimate the viewpoint of the objects, is a crucial feature in order to project a realistic hologram, for instance. For the ITS, the capability of detecting the objects and estimating their viewpoints results also fundamental. For example, localizing and estimating the pose of other vehicles are essential tasks for the autonomous navigation; or in the vehicle tracking and monitoring task in highways, adding pose information improves the system performance (Guerrero-Gómez-Olmedo et al. 2013).

Technically, given an image, these models can localize the objects, predicting their associated bounding boxes, and are also able to estimate the relative pose of the object instances in the scene with respect to the camera. Figure 3.1 shows some examples, where the viewpoint of the object is encoded using just the azimuth angle. In the image. The locations of the objects are depicted by their bounding boxes (in green), and their azimuth angles are represented by the blue arrow inside the yellow circle.

The computer vision community rapidly detected the necessity of providing the appropriately annotated datasets, in order to experimentally validate the object detection and pose estimations approaches. To date, several datasets have been released. Some examples are: 3D Object categories (Savarese 2008), EPFL Multi-view car (Ozuysal et al. 2009), ICARO (López-Sastre et al. 2010), PASCAL3D+ (Xiang et al. 2014) or ObjectNet3D (Xiang et al. 2016).

Thanks to these datasets, multiple models have been experimentally evaluated. It is particularly interesting to observe how all the published approaches can be classified into two groups. In the first one, we find those models that decouple both problems (*e.g.* (Tulsiani and Malik 2015), (Glasner et al. 2012), and (Redondo-Cabrera and Lopez-Sastre 2015)), making first a location of the object, to later estimate its pose. In the second group,



(a) (b)

(c)

Figure 3.1: Object category detection and pose estimation example. In the images, the objects are localized by the green bounding boxes. The blue arrow inside the yellow circles shows the azimuth angles of the objects, which is a form of viewpoint annotation.

we identify the approaches that solve both tasks simultaneously (*e.g.* (Xiang et al. 2014), (Pepik et al. 2012), and (Massa et al. 2016)), because they understand that to carry out a correct location requires a good estimation of the pose and vice versa.

But the discrepancies do not end here. Unlike the problem of object detection, where the metric for the experimental evaluation is clear, being this the mean Average Precision (mAP) defined in the PASCAL VOC Challenge, for the problem of object detection and pose estimation, multiple metrics have been adopted. This is motivated by the fact that not all the models understand the viewpoint estimation problem in the same way. Some solutions, *i.e.* the discrete approaches, consider that this is a classification problem, when others, *i.e.* the continuous models, understand the pose as a continuous variable, whose estimation must be approached by solving a regression problem.

This chapter is an attempt to provide a comparative study where these issues can be addressed. The main contributions are as follows:

- We introduce three novel deep learning architectures for the problem of simultaneous object detection and pose estimation. Our models seek to perform a joint detection and pose estimation, trained fully end-to-end. We start with a model that fully integrates the tasks of object localization and object pose estimation. Then, we present two architectures that gradually decouple both tasks, proposing a final deep network where the integration is minimal. All our solutions are detailed in Sections 3.3.1 and 3.3.2.

- All our architectures have been carefully designed to be able to treat the pose estimation problem from a continuous or from a discrete perspective. We simply need to change the loss functions used during learning. This is detailed in Section 3.3.3. In our experiments, we carefully compare the performance of these two families of methods, reporting results using four different loss functions. Therefore, this thesis aims to shed some light on which perspective is more appropriate, keeping the network architecture fixed.

- Thanks to the proposed models, we are able to offer an experimental evaluation (see Section 3.4) designed to carefully analyze how coupled the detection and pose estimation tasks are, being this our final contribution. We also bring a detailed comparison with all the solutions that establish the state of the art for the problem of object category detection and pose estimation. We carefully analyze all the models using two publicly available datasets: PASCAL3D+ (Xiang et al. 2014) and ObjectNet3D (Xiang et al. 2016).

3.2 Related work

The literature on 3D object pose estimation is large. In the early days of the field, we mostly find works that exploit the modeling of an object shape and scene geometry to estimate the pose. In the work of Harris and Stennett

(1990), they proposed a model-based tracking algorithm for a known three-dimensional object. In their work, they match the edges and the corners of 2D images with the edges and corners of an approximated 3D model. Afterward, they track the corner points and edges by using a Kalman filter. Later on in the computer vision community, the local descriptors such as SIFT (Lowe 1999), SURF (Bay et al. 2006), ORB (Rublee et al. 2011) and others got a lot of attention. The method of Lowe (2004) presents a mechanism for extracting distinctive invariant features from images that are used to perform a reliable matching between different views of an object or scene. In Figure 3.2 is shown some qualitative results extracted from the aforementioned works.

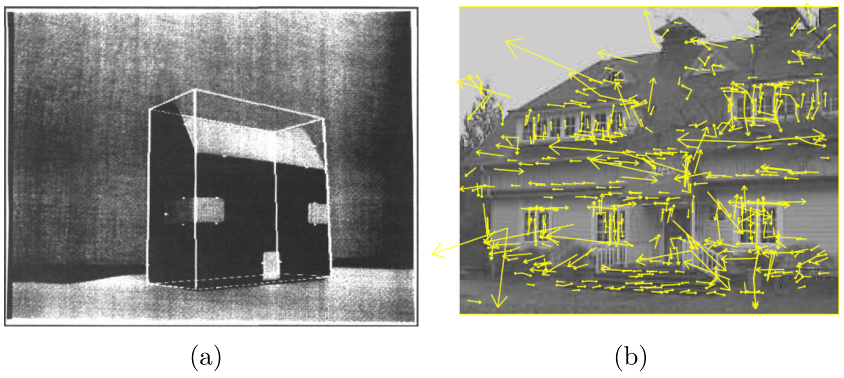
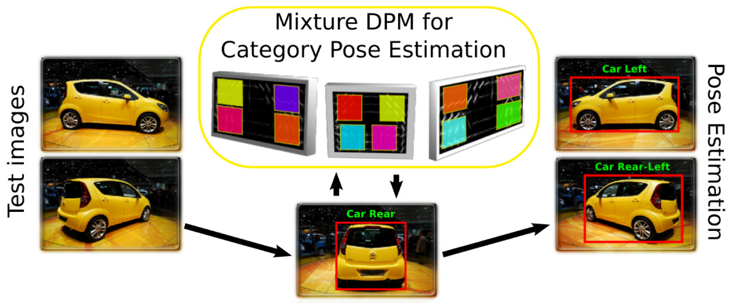


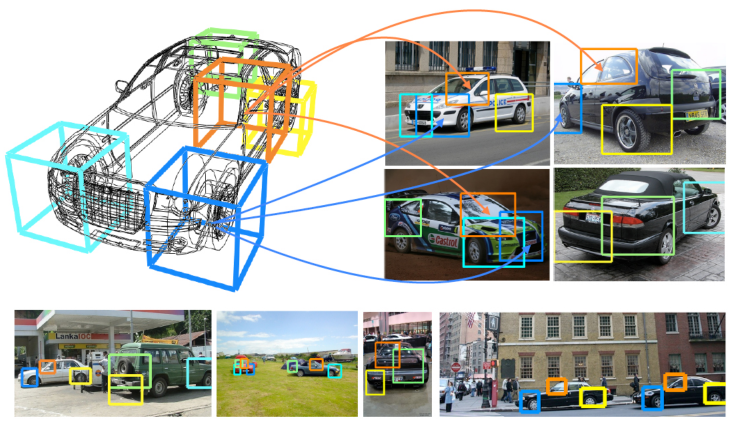
Figure 3.2: (a) and (b) show the qualitative results reported on the original works of Harris and Stennett (1990) and Lowe (2004) respectively.

Alternatively to the point matching and geometry bases approaches, machine learning is one of the most important techniques. In contrast with the previous solutions, the machine learning aims to learn a general function that automatically maps the appearance of the target object to its pose based on a collection of data samples. Based on machine learning techniques, we find the VDPM of Lopez-Sastre et al. (2011). In their work, they discriminatively trained a Deformable Part Model (DPM) such that each mixture component represents a different azimuth section used to estimate the pose along with the object detection. In the work of Pepik et al. (2012), they utilize structural SVM, to predict the object bounding box and pose jointly (later on referenced as DPM-VOC+VP). In Figure 3.3 are depicted the system diagrams of the VDPM in (a), and of the DPM-VOC+VP in (b).

With the success of deep learning, a totally new branch of methods appeared. Following this trend, we find the pioneering work of Tulsiani and Malik (2015). In their work they use the RCNN (Girshick et al. 2014) for detecting the objects. They fine-tune the VGG16 (Simonyan and Zisserman 2014) on the detection crops to predict the discrete pose. Beyer et al. (2015)



(a)



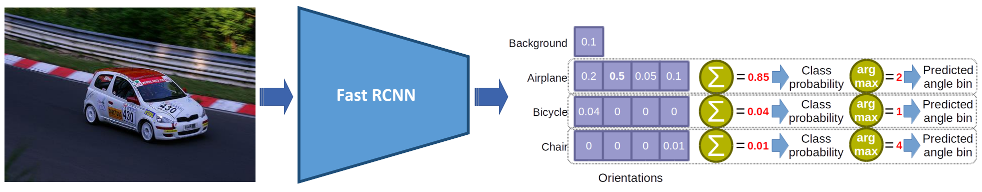
(b)

Figure 3.3: (a) shows the diagram presented in the original paper of Lopez-Sastre et al. (2011), and (b) show the diagram presented on the work of Pepik et al. (2012).

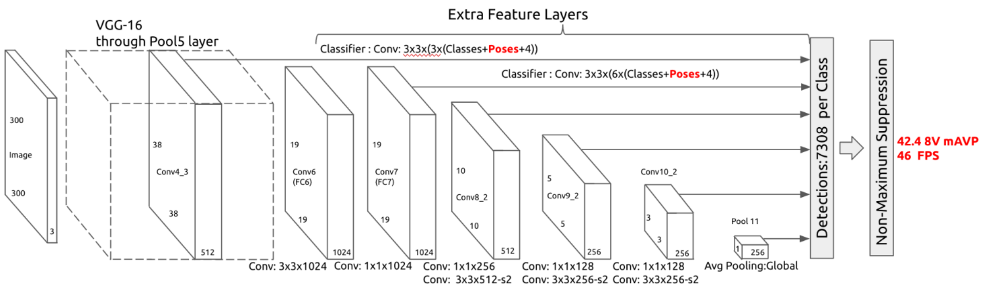
they train a CNN to perform a continuous pose estimation where the pose is decomposed in polar coordinates. In the work of Massa et al. (2016), they modify the Fast RCNN (Girshick 2015) to simultaneously perform the object detection and pose estimation, but in contrast with the Fast RCNN, the class of the object is determined by the top accumulated score over the predicted pose for a certain object class. Poirson et al. (2016) extend YOLO (Redmon et al. 2016) to include the pose estimation together with the object detection. Figure 3.4 shows the approach diagram introduced in (Massa et al. 2016), and (Poirson et al. 2016).

It is clear that object category detection and viewpoint estimation is a growing research field. Several are the methods that have contributed to improving the state of the art. Like we have previously mentioned, we can organize in two groups all the approaches in the literature.

In the first one, we find those models that understand that these two



(a)



Extra Feature Layers
VGG-16 through Pool5 layer
Classifier : Conv: 3x3x(3x(Classes+Poses+4))
Classifier : Conv: 3x3x(6x(Classes+Poses+4))
Image
Conv4_3
Conv6 (FC6)
Conv7 (FC7)
Conv8_2
Conv9_2
Conv10_2
Pool 11
Conv: 3x3x1024
Conv: 1x1x1024
Conv: 1x1x256 Conv: 3x3x512-s2
Conv: 1x1x128 Conv: 3x3x256-s2
Conv: 1x1x128 Conv: 3x3x256-s2
Avg Pooling:Global
Detections:7308 per Class
Non-Maximum Suppression
42.4 8V mAVP
46 FPS

(b)

Figure 3.4: (a) depicts the system diagram of Massa et al. (2016). (b) shows the the diagram proposed by Poirson et al. (2016).

tasks, *i.e.* object localization and pose estimation, must be solved separately ((Tulsiani and Malik 2015), (Glasner et al. 2012), and (Redondo-Cabrera and Lopez-Sastre 2015)). The second group consists of the models where the detection and the viewpoint estimation are fully coupled ((Xiang et al. 2014), (Pepik et al. 2012), (Massa et al. 2016), and (Redondo-Cabrera et al. 2014)).

Within these two groups, one must note that while some models solve the pose estimation as a classification problem, *i.e.* the discrete approaches ((Massa et al. 2016), and (Tulsiani et al. 2015)), others treat the viewpoint estimation as a regression problem, *i.e.* the continuous solutions ((Glasner et al. 2012), (Redondo-Cabrera et al. 2014), and (Fenzi et al. 2013)).

In this work, we introduce three novel deep learning architectures for the problem of *joint* object detection and pose estimation. They all are extensions for the Faster R-CNN object detection model of Ren et al. (2015). We have designed them to gradually decouple the object localization and pose estimation tasks. Our models significantly differ from previous deep learning based approaches for the same tasks. For instance, if we consider the work of Tulsiani et al. (2015), we observe that their solution is based on a detector (using the R-CNN (Girshick et al. 2014)), followed by a pose classification network, fully decoupling both tasks. On the contrary, all our architectures are trained fully end-to-end, performing a joint detection and viewpoint estimation. Moreover, the deep architectures implemented are different. Massa et al. (2016) also propose a *joint* model. However, their approach is completely different. They base their design on the Fast R-CNN detector (Girshick 2015). Technically, they modify the Fast R-CNN output to provide the detections based on an accumulative sum of scores that is provided by the pose classification for each object category. In a different manner, our solutions are based on the Faster R-CNN, which is a distinct architecture. Moreover, in our work, we explore not only a modification of the output of the networks, but multiple architecture designs where we can gradually separate the branches of the network dedicated to the object localization and the viewpoint estimation tasks.

Finally, this works offers a detailed comparative study of solutions for the joint object *detection* and pose estimation problem. The study included in (Elhoseiny et al. 2016) focus on the different problem of object *classification* and pose estimation, *i.e.* they do not consider the object localization task.

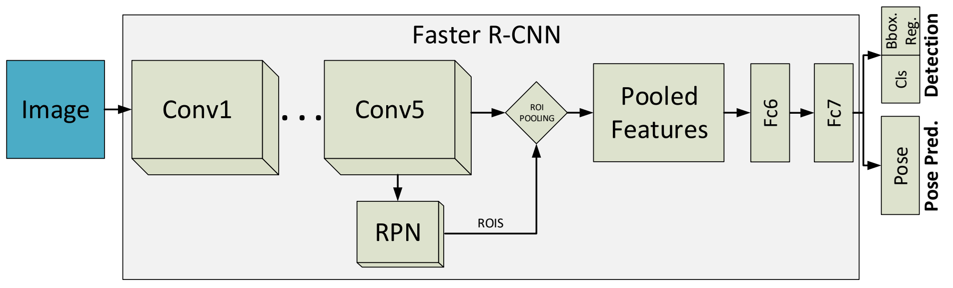
3.3 Simultaneous detection and pose estimation models

In the following section, we formulate the learning problem for joint detection and pose estimation. Then, we detail the proposed architectures, named: single-path, specific-path, and specific-network (Figure 3.5 shows an overview of all our designs). Technically, they all are extensions for the Faster R-CNN approach (Ren et al. 2015). Finally, we provide a detailed analysis of the loss functions used in our experimental evaluation.

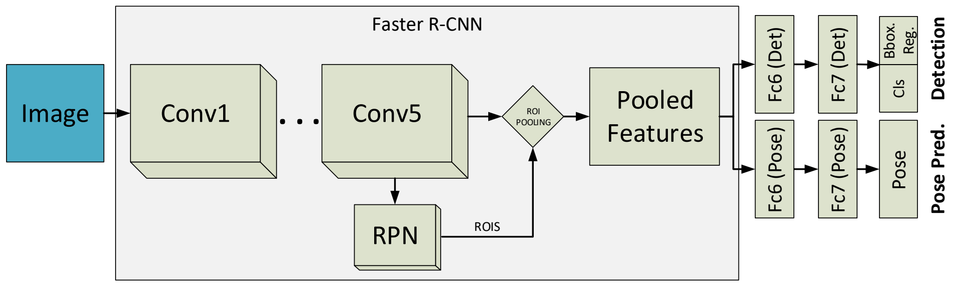
3.3.1 Learning model for simultaneous detection and pose estimation

Our goal is to learn a strong visual representation that allows the models to: localize the objects, classify them and estimate their viewpoint with respect to the camera. Furthermore, we consider an *in the wild* setting where multiple objects of a variety of categories appear in real-world scenarios, with a considerable variability on the background, and where occlusions and truncations are the rules rather than the exception.

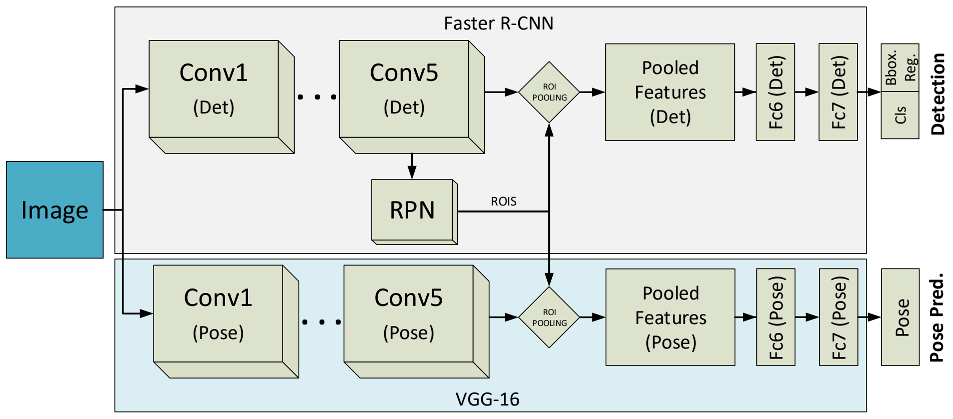
Therefore, the supervised learning process starts from a training set $S = \{(x_i, t_i)\}_{i=1}^{N}$, where N is the number of training samples. For each sample i in the dataset, $x_i \in X$ represents the input image, and $t_i \in T$, with $t_i = (y_i, \beta_i, \phi_i)$, encodes the annotations for the three tasks to solve: classification



(a) Single-path architecture.



(b) Specific-path architecture.



(c) Specific-network architecture.

Figure 3.5: Proposed deep learning architectures for simultaneous object detection and pose estimation.

(y_i), object localization (β_i) and pose estimation (ϕ_i). $y_i \in Y$ with $Y = [1, 2, \dots, C, C+1]$ describes the object class, being C the total number of object categories. Category $C+1$ is used to consider a generic background class. $\beta_i \in \mathbb{R}^4$ represents the bounding box localization of a particular object within image x_i. Finally, $\phi_i \in \mathbb{R}^3$ encodes the 3D viewpoint annotation for a particular object with respect to the camera position as a tuple of azimuth, elevation and zenith angles.

We propose to learn a convolutional neural network (CNN) (LeCun et al. 1990) for simultaneous object detection and pose estimation. Technically, these CNNs are a combination of three main features which let the model achieve a sort of invariance with respect to imaging conditions: local receptive fields, shared convolutional weights, and spatial pooling. Each unit in a layer receives inputs from a set of units located in a small neighborhood of the previous layer. In the forward pass of a CNN, each output feature is computed by the convolution of the input feature from the previous layer. Therefore, these deep networks can be thought of as the composition of a number of convolutional structure functions, which transform the input image to feature maps that are used to solve the target tasks.

For the particular problem of simultaneous object detection and viewpoint estimation, our CNN prediction $\hat{t}$ should be expressed as follows,

$$\hat{t}_{\boldsymbol{\theta},W} = F_W \circ \mathbf{z}_\theta(x_i)\,. \tag{3.1}$$

$\mathbf{z}_\theta : X \to \mathbb{R}^D$ represents the D-dimensional feature mapping that the network performs to the input images. Technically, it consists in the transformation of the input image x_i into a feature that is used to feed the output layers of our models. We encode in $\boldsymbol{\theta}$ the trainable weights of the deep architecture that allow the network to perform the mapping. In our solutions, the weights in $\boldsymbol{\theta}$ define the hidden layers that are shared by all the tasks that the deep network needs to solve.

F_W corresponds to the set of functions of the output layers. They take as input the deep feature map $\mathbf{z}_\theta(x_i)$. For the problem considered in this paper, our set of functions must address three different tasks: classification (y), object localization (β) and viewpoint estimation (ϕ). Therefore, $F_W = (f^y_{W^y}, f^\beta_{W^\beta}, f^\phi_{W^\phi})$. $f^y_{W^y}$ with weights W^y produces the prediction for the object category, *i.e.* $\hat{t}^y$. $f^\beta_{W^\beta}$ predicts the object location $\hat{t}^\beta$. Finally, $f^\phi_{W^\phi}$ is in charge of the prediction of the viewpoint $\hat{t}^\phi$.

According to the prediction model detailed in Equation 3.1, we define the following objective function to learn our multi-task neural network:

$$\underset{\boldsymbol{\theta},W}{\operatorname{argmin}}\, \mathcal{L}(\boldsymbol{\theta}, W, S)\,, \tag{3.2}$$

where the loss function follows the equation,

$$\mathcal{L}(\boldsymbol{\theta}, W, S) = \lambda_1 \mathcal{L}_y(\boldsymbol{\theta}, W^y, S) + \lambda_2 \mathcal{L}_\beta(\boldsymbol{\theta}, W^\beta, S) + \lambda_3 \mathcal{L}_\phi(\boldsymbol{\theta}, W^\phi, S)\,. \quad (3.3)$$

λ_i for $i \in (1, 2, 3)$ corresponds to the scalar value that controls the importance of a particular loss during training. For the classification loss $\mathcal{L}_y$ we use a categorical cross-entropy function. A simple Euclidean loss is used for the object localization task loss $\mathcal{L}_\beta$. Finally, for the pose estimation loss $\mathcal{L}_\phi$ multiple options are considered. We detail them in Section 3.3.3.

3.3.2 The proposed architectures

3.3.2.1 Single-path architecture

Our first deep network design is the *single-path* architecture. It offers a natural extension of the Faster R-CNN model for the problem of simultaneous object detection and pose estimation. Technically, we simply add an extra output layer in order to predict the viewpoint of the object.

To understand the extension proposed, we proceed with a description of the original Faster R-CNN pipeline. As it is shown in Figure 3.5(a), the Faster R-CNN consists of three stages. The first stage is performed by the convolutional layers. An input image passes through the convolutional network, to be transformed into a deep feature map. The second stage is represented by the Region Proposal Network (RPN), which serves as an “attention” mechanism during learning. Technically, it is a fully convolutional (sub)network, which takes an image feature map as input, and outputs a set of rectangular object proposals, with their corresponding objectness scores. To go into details, this RPN takes the feature map obtained from the last convolutional layer (*e.g.* convolution 5 in a VGG16-based architecture) and adds a new convolutional layer which is in charge of learning to generate regions of interest (ROIs). In the third stage, these ROIs are used for pooling those features that are passed to the last two fully-connected (FC) layers. Finally, the responses coming from the last FC layer are used by the model: 1) to classify the ROIs into background or object; 2) to perform a final bounding box regression for a fine-grained localization of the object. In Figure 3.5(a) we represent these two tasks with the blocks named as “Cls” (for classification) and “Bbox. Reg.” (for the bounding box regression). Technically, the “Cls” module is implemented with a softmax layer, and the “Bbox. Reg.” layer is a linear regressor for the four coordinates that define a bounding box.

In order to evaluate the capability of the Faster R-CNN for the task of pose estimation, guaranteeing a minimal intervention in the model architecture, we propose the *single-path* extension. It consists in the incorporation of an additional output layer (see box "Pose" in Figure 3.5(a)), connected to the last FC layer as well. The objective of this layer is to cast a prediction for the viewpoint, and to measure the loss for this task, propagating the appropriate gradients to the rest of the network during learning.

For training this *single-path* model, we solve the objective loss function of Equation 3.2. We give the same weight to each task, *i.e.* $\lambda_1 = \lambda_2 = \lambda_3 = 1$. Note that at this point, we do not specify whether the viewpoint estimation will be considered as a classification or regression problem. In this sense, different loss functions will be considered and evaluated in the experiments, in order to attain a high level of understanding of the simultaneous detection and pose estimation problem.

3.3.2.2 Specific-path architecture

The *specific-path* is our second approach. Our objective with this architecture is to explore the consequences of a slight separation of the pose estimation task from the object class detection, learning *specific deep features* for each task.

As it is shown in Figure 3.5(b), the extension we propose for this second approach consists of adding two independent FC layers, which are directly connected to the pose estimation layer. Note that we do not change the rest of the architecture, *i.e.* both the initial convolutional layers and the RPN module are shared. The pooled features are used to feed the two groups of FC layers that form two types of features: one for the object detection task, and the other for the viewpoint estimation. Therefore, during training, each network FC path learns its specific features based on its gradients, while the rest of the layers learn a shared representation.

The model is learned solving the objective function shown in Equation 3.2. For the detection path, $\lambda_1 = \lambda_2 = 1$, and $\lambda_3 = 0$. For the pose path we solve the Equation 3.2 getting rid of the object classification and bounding box regression losses, *i.e.* $\lambda_1 = \lambda_2 = 0$ and $\lambda_3 = 1$.

3.3.2.3 Specific-network architecture

With our third architecture, named *specific-network*, we attempt to separate as much as possible the detection and pose estimation tasks within the same architecture. The key idea of this design is to provide a model with two networks that can be fully specialized in their respective tasks, while they are learned simultaneously and end-to-end.

Consequently, as it is shown in Figure 3.5(c), we design a model made of two independent networks: the *detection network* and the *pose network*. The *detection network* is in charge of fully performing the object localization task, as in the original design of the Faster R-CNN.

The *pose network* must focus on the viewpoint estimation task, without any influence of the detection objective. Therefore, this network has now its own initial convolutional layers. To align the detection and pose estimation, the *pose network* receives the ROIs generated by the RPN module of the *detection network*. Technically, an input image is forwarded simultaneously into both convolutional networks. The second stage of the Faster R-CNN, *i.e.* the generation of ROIs by the RPN, occurs in the detection network only. These ROIs are shared with the *pose network*. Finally, each network pools its own features from the generated ROIs, feeds its FC layers with these features, and produces its corresponding outputs.

Overall, we have an architecture with two specialized networks, that are synchronized to solve the object detection and pose estimation tasks in a single pass.

For learning this model we follow the same procedure as for the *specific-path*. We train our *detection network* to solve the Equation 3.2 where $\lambda_3 = 0$ and $\lambda_1 = \lambda_2 = 1$. The *pose network* is solved just for the pose problem, hence, $\lambda_1 = \lambda_2 = 0$ and $\lambda_3 = 1$. The main difference with respect the *specific-path* model is that there are no shared features, so each network is fully specialized to solve its corresponding task.

3.3.2.4 Why have we chosen these designs?

All our architectures are extensions of the Faster R-CNN approach of Ren et al. (2015). Originally, the Faster R-CNN architecture was proposed to address the problem of object detection only. This model has systematically prevailed on all the detection benchmarks (*e.g.* PASCAL VOC (Everingham et al. 2010), COCO (Lin et al. 2014) and ILSVRC detection (Deng et al. 2009)), where leading results are obtained by Faster R-CNN based models, albeit with deeper features (*e.g.* using deep residual networks (He et al. 2016)). So, following a simple performance criterion, we believe that the Faster R-CNN with its excellent results is a good choice.

Our second criterion for the selection of this Faster R-CNN architecture is related to the main objective of our research: propose and evaluate solutions for the problem of *simultaneous* object detection and viewpoint estimation. Note that we neither address the problem of pose estimation in a classification setup in isolation (*e.g.* in (Elhoseiny et al. 2016), where the object localization problem is not considered) nor decouple the object detection and pose estimation tasks (*e.g.* the work of Tulsiani et al. (2015)). Our models seek to perform a *joint* detection and pose estimation, trained fully end-to-end, and the Faster R-CNN architecture is an ideal candidate to extend. All our solutions perform a direct pooling of regions of interests in the images from the internal RPN of the Faster R-CNN. This way, we do not need to use any external process to hypothesize bounding boxes (*e.g.* Selective Search (Uijlings et al. 2013)), hence performing a truly end-to-end simultaneous object detection and pose estimation model, where the weights of the fully convolutional RPN learn to predict object bounds and objectness scores at each position, to maximize not only the object detection accuracy but also the viewpoint estimation performance.

Finally, we want to discuss our main arguments for the concrete extensions proposed in our architectures. Traditionally, the computer vision community working on the problem of pose estimation for object categories has been divided into two groups. Those that understand that the tasks of localizing objects and estimating their poses are decoupled tasks (*e.g.* (Tulsiani et al. 2015), (Glasner et al. 2012), (Fenzi et al. 2013), and (Redondo-Cabrera et al. 2014)), and those that advocate for jointly solving both tasks (*e.g.* (Massa et al. 2016), (Su et al. 2015), (Redondo-Cabrera et al. 2014), and (Pepik et al. 2012)). The architectures proposed in this paper move from a *fully* integration of both tasks, *i.e.* in the *single-path*, towards the *specific-network* model, where the integration is minimal. In this way, we can design an experimental evaluation to thoroughly analyze how coupled the detection and pose estimation tasks are. Moreover, all our experiments are carried on publicly available datasets which have been designed for the problem of detection and viewpoint estimation, therefore a direct comparison with previous methods that define the state of the art is also possible.

3.3.3 Loss functions for pose estimation

Unlike the well-defined object detection task, the viewpoint estimation problem has been traditionally considered from two different perspectives: the continuous and the discrete. Most methods in the literature adopt the discrete formulation. That is, they understand the pose estimation as a classification problem, relying on a coarse quantization of the poses for their multi-view object detectors (*e.g.* (Tulsiani and Malik 2015), (Pepik et al. 2012), and (Su et al. 2015)). Only a few approaches consider that the pose estimation of categories is ultimately a continuous problem, *i.e.* a regression problem (*e.g.* (Redondo-Cabrera et al. 2014), (Fenzi et al. 2013), and (Beyer et al. 2015)). In this chapter, all our architectures are evaluated considering these two perspectives for the viewpoint estimation.

When we want our models to consider discrete outputs for the pose estimation (the "Pose" layer in Figure 3.5), we integrate the following categorical cross-entropy loss function in Equation 3.2:

$$\mathcal{L}_\phi(\boldsymbol{\theta}, W^\phi, S) = -\frac{1}{N}\sum_{i=1}^{N} \log\left(\sigma_{l_i^\phi}(f_{W^\phi}^\phi \circ \mathbf{z}_\theta(x_i)\right) \,, \tag{3.4}$$

where N is the number of samples, and σ_{l^ϕ} is the softmax function for the label l_i^ϕ.

When the pose estimation is considered from a continuous perspective, multiple adequate regression loss functions can be integrated. For all them, it is fundamental to deal with the circularity of the viewpoint. Therefore, we first represent the orientation angles as points on a unit circle by the following transformation, $\mathbf{p}(\alpha) = (\sin(\alpha), \cos(\alpha))$, $\mathbf{p}(\alpha) \in \mathbb{R}^2$.

Probably, the simplest way to train the pose regressor is by using an Euclidean loss, as follows:

$$\mathcal{L}_\phi(\boldsymbol{\theta}, W^\phi, S) = \frac{1}{2N}\sum_{i=1}^{N} \left(\mathbf{p}\left(l_i^\phi\right) - \mathbf{p}\left(f_{W^\phi}^\phi \circ \mathbf{z}_\theta(x_i)\right)\right)^2 \,. \tag{3.5}$$

A popular alternative to the Euclidean loss, is the Huber loss function,

$$\mathcal{L}_\phi(\boldsymbol{\theta}, W^\phi, S) = \tfrac{1}{N}\textstyle\sum_{i=1}^{N} \begin{cases} \frac{1}{2}\left(\mathbf{p}\left(l_i^\phi\right) - \mathbf{p}\left(f_{W^\phi}^\phi \circ \mathbf{z}_\theta(x_i)\right)\right)^2 & \text{if } \left|\mathbf{p}\left(l_i^\phi\right) - \mathbf{p}\left(f_{W^\phi}^\phi \circ \mathbf{z}_\theta(x_i)\right)\right| \leq \delta, \\ \delta\left|\mathbf{p}\left(l_i^\phi\right) - \mathbf{p}\left(f_{W^\phi}^\phi \circ \mathbf{z}_\theta(x_i)\right)\right| - \frac{1}{2}\delta^2 & \text{otherwise} \end{cases} . \tag{3.6}$$

The advantage of this loss is that it tends to be more robust to outliers than the Euclidean loss.

Finally, we propose to also use the continuous cyclic cosine cost function, which is widely used in the natural language processing literature (Singhal 2001). It is defined as follows,

$$\mathcal{L}_\phi(\boldsymbol{\theta}, W^\phi, S) = \frac{1}{N}\sum_{i=1}^{N} \left(1 - \frac{\mathbf{p}(l_i^\phi)\mathbf{p}(f_{W^\phi}^\phi \circ \mathbf{z}_\theta(x_i))}{\left\|\mathbf{p}(l_i^\phi)\right\| \left\|\mathbf{p}(f_{W^\phi}^\phi \circ \mathbf{z}_\theta(x_i))\right\|}\right) . \tag{3.7}$$

3.4 Experiments

3.4.1 Implementation details

To perform our experiments, we have implemented all our models and loss functions using the deep learning framework Caffe (Jia et al. 2014b). The optimization is done by using the Stochastic Gradient Descent algorithm, with: a momentum of 0.9; a weight decay of 0.0005; and a learning rate of 0.001. The learning rate of the output layer for the pose estimation has been

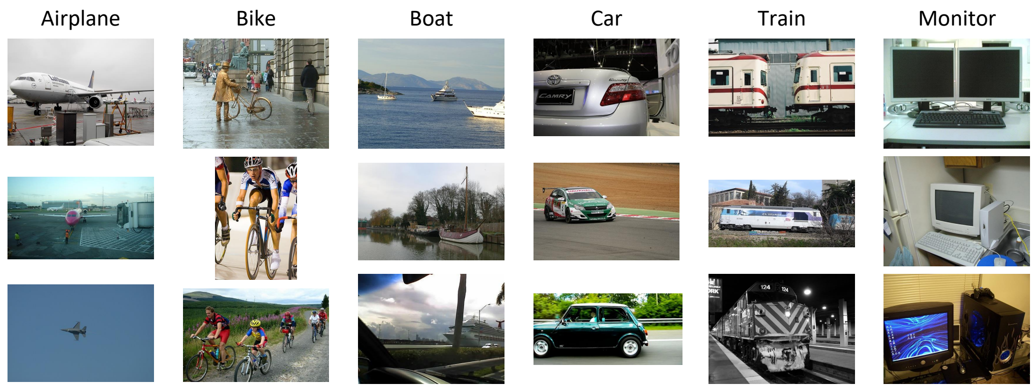


Figure 3.6: Some images of the PASCAL3D+ dataset.

multiplied by a factor of 0.01, so as to guarantee that the network properly converges. We publicly release all our implementations[1].

We follow the standard procedure of the Faster R-CNN of Ren et al. (2015) for training the models in an end-to-end fashion. This way, for each training iteration, just one image is taken and passed through the first set of convolutions. In a second step, a collection of 128 region proposals is generated. These regions are used to build the batch to feed the last set of FC layers. This batch contains 32 samples of foreground samples and 96 samples of background.

For the experimental evaluation, we use two publicly available datasets, which have been specially designed for the evaluation of object detection and pose estimation models: PASCAL3D+ (Xiang et al. 2014) and ObjectNet3D (Xiang et al. 2016). We strictly follow the experimental setup described in these datasets. In the following sections, more details are provided, as well as a thorough analysis of the results and main conclusions obtained.

3.4.2 Results in the PASCAL3D+ dataset

PASCAL3D+ (Xiang et al. 2014) dataset is one of the largest and most challenging datasets for the problem of object detection and poses estimation. Technically, it consists of:

- The images and annotations of the 12 rigid object categories provided with the PASCAL VOC 2012 dataset (Everingham et al. 2010).

[1]The source repository is available for the public, and it can be found at `https://github.com/gramuah/pose-estimation-study.git`

- An additional set of 22,394 images taken from the ImageNet (Deng et al. 2009) dataset, for the same 12 categories.

On average, it has more than 3000 instances per object category. The test set has 5823 images directly inherited from the PASCAL VOC 2012 test subset. Figure 3.6 shows some examples of images. One can clearly observe that the images provided contain objects "in the wild". The standard PASCAL VOC annotation for all the objects (*i.e.* category label and bounding box), has been extended to provide a precise 3D pose. This has been done performing a manual alignment of the objects in the images with 3D CAD models. This way, azimuth, elevation, and distance from the camera pose in 3D are provided for each object.

For our analysis, we follow the official experimental setup of the PASCAL3D+ (Xiang et al. 2014). The evaluation metric for the object detection and pose estimation is the Average Viewpoint Precision (AVP). This AVP is similar to the Average Precision (AP) for object detection. To compute the AVP, every output of the detector is considered to be correct if and only if the bounding box overlap with the ground truth annotation is larger than 50% *and* the viewpoint estimation for the azimuth angle is correct. When we consider a discrete space for the viewpoint, the viewpoint estimation is correct if it coincides with the ground truth azimuth label. On the contrary, if the viewpoint belongs to a continuous space, then, two viewpoint labels are correct if the distance between them is smaller than a fixed threshold of $\frac{2\pi}{v}$, where v is the number of views.

3.4.2.1 Network initialization analysis

One of the most common practices in deep learning consists in initializing a deep network architecture with the weights of a model pre-trained in a big dataset, such as ImageNet (Deng et al. 2009), and then start a fine-tuning process for a specific task, typically using a different dataset.

For our problem of joint object detection and pose estimation, we also follow this popular recipe. In a nutshell, we fine-tune our networks in the PASCAL3D+ dataset, using for the initialization of the weights two pre-trained models: the original VGG16 model (Simonyan and Zisserman 2014) trained for the ImageNet dataset, and the Faster R-CNN model (Ren et al. 2015) using only the training set of the PASCAL VOC 2012 dataset. Note that the validation set of the original PASCAL VOC 2012 is now the test set proposed in the PASCAL3D+, therefore, we do not allow the Faster R-CNN to be pre-trained on it. For the rest of the model weights that are not covered by the pre-trained models, we basically follow a standard random initialization.

Here we simply want to explore what initialization procedure is the best option. Therefore, for this preliminary experiment, we just use our first architecture, the *Single-path*. The pose estimation is considered as a classification problem, using 360 discrete bins, and we employ the cross-entropy loss defined in Eq. 3.4.

Init. strategy	mAP	mAVP 4	mAVP 8	mAVP 16	mAVP 24
ImageNet	49.5	37.6	32.0	**24.6**	**20.2**
PASCAL VOC 2012	**63.6**	**42.4**	**32.2**	23.6	18.9

Table 3.1: Effect of the network initialization strategy in the PASCAL3D+ for the *Single-path* architecture.

Table 3.1 shows the main results using the described initialization strategies. In terms of object detection precision, *i.e.* mAP, the initialization of our model, using the PASCAL VOC 2012 dataset is the best option, by a considerable margin, with respect to the ImageNet based strategy. Interestingly, the mAP of our model (63.6) improves the state of the art for the object detection task in the official PASCAL3D+ leaderboard [2], where the best mAP is of 62.5 reported by Massa et al. (2016).

In terms of joint object detection and pose estimation, we also report the mAVP for different sets of views (4, 8, 16 and 24). The ImageNet based initialization reports slightly better results only for the more fine-grained setups of 16 and 24 views. When just 4 or 8 views are considered, the initialization process using the PASCAL VOC 2012 is the best option, considering its high detection precision. This first experiment also reveals that it seems to be a trade-off between how good the system is localizing objects and how accurate the pose predictions are. Overall, we conclude that the best initialization strategy is clearly the one based on the PASCAL VOC 2012 dataset. Therefore, for the rest of the experiments, we follow this initialization strategy.

3.4.2.2 Discrete vs. Continuous approaches analysis

As we have discussed in Section 3.3.3, the pose estimation problem can be treated following either a discrete approach, *i.e.* as a classification problem, or a continuous approximation, *i.e.* as a regression problem. One of the main objectives of our study is to shed light on this discussion.

[2]Official PASCAL3D+ leaderboard is available at `http://cvgl.stanford.edu/projects/pascal3d.html`

Losses	mAP	mAVP 4	mAVP 8	mAVP 16	mAVP 24
Discrete (Eq. 3.4)	63.6	42.4	32.2	**23.6**	**18.9**
Euclidean (Eq. 3.5)	64.3	**47.9**	**34.7**	23.2	17.6
Huber (Eq. 3.6)	**64.5**	46.1	31.5	20.2	15.2
Cyclic Cosine (Eq. 3.7)	55.6	42.1	32.2	22.5	17.5

Table 3.2: Loss function analysis for the PASCAL3D+ dataset. Object detection and viewpoint estimation performances are reported.

We have carefully designed all our architectures, so they all can consider a discrete and a continuous approximation to the pose estimation problem. We simply have to *change* the Pose estimation layer and its associated loss function. Up to four different loss functions are analyzed in these experiments, one for the discrete case and three for the continuous approach.

When the discrete scenario is considered, we follow the cross-entropy loss function in Equation 3.4. Technically, our architectures consider 360 different classes for the azimuth angle. For each category in the dataset (except for the background), we learn a specific pose estimator, therefore, we need to define a softmax function with a length of $360 \times C$ elements, where C is the number of classes. During learning, we have opted to *mask* the softmax layer, propagating only the error for the elements that correspond to the pose of the foreground class.

For the continuous pose estimation problem, our networks learn to directly perform the regression of the two values corresponding to the conversion to polar coordinates the azimuth angle. We design our deep models to learn a particular regressor for each object category. And again, during learning, only the regressor that corresponds to the associated class label of the sample in the training batch is allowed to propagate errors. Following this continuous setup, we analyze the three different loss functions introduced in Section 3.3.3: the Euclidean loss (Eq. 3.5), the Huber loss (Eq. 3.6), and the Cyclic cosine loss (Eq. 3.7).

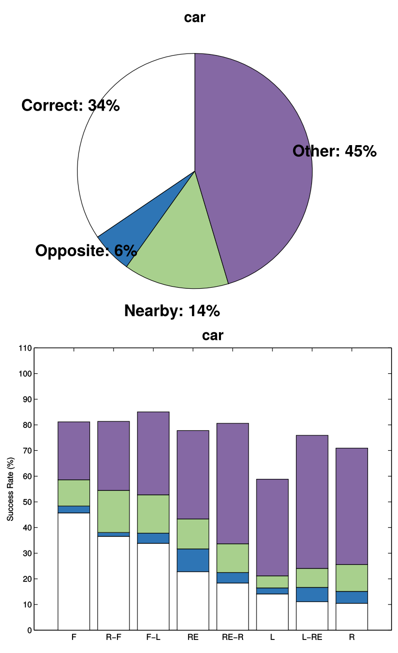
Table 3.2 shows the main results when the different loss functions are used. Discrete, Euclidean and Huber losses exhibit a very similar detection performance (mAP). Only when the Cyclic Cosine loss is used, a substantial drop in the detection performance is reported. The reason we find to explain this fact is that during training, the Cyclic Cosine loss can eventually produce larger gradients than the detection loss. This issue causes that the learning process tends to focus more “attention” on the pose estimation task, obtaining a deep model with a worse object localization accuracy. A simple adjustment of the λ values in Eq. 3.3 did not properly work in our experiments. Another possibility could be to perform a power normalization of the gradients produced by the different losses at the same level of the network. However, we did not explore this option. Instead, we opted for applying the clipping gradient strategy (Pascanu et al. 2013), with a threshold value of 5.

If we analyze now the mAVP, where both object detection and viewpoint estimation accuracies are considered, we can observe that, in general, the best performance is reported when the Euclidean loss based model is used. Moreover, within the group of continuous viewpoint estimation models, the Euclidean is the clear winner. Therefore, for the rest of the paper, when a continuous viewpoint model is learned, we use the Euclidean loss. Interestingly, the continuous approach wins the discrete model only when 4 and 8 set of views are considered. For 16 and 24 views, the discrete model retrieves a slightly better performance. In our experiments, we have noted that the continuous pose estimation approaches tend to offer smooth predictions that are concentrated around the most frequent viewpoint of the training set. However, the discrete approach, with a Softmax loss, does not suffer that much from this pose annotation bias.

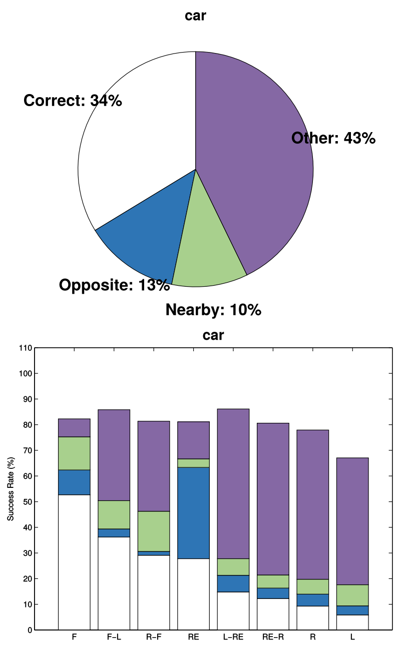
Figure 3.7 shows a detailed comparison of the performance between a discrete and a continuous approach for a pair of representative object categories in the context of ITS solutions: *car* and *bus*. *Car* is the class with the largest amount of samples in the PASCAL3D+ dataset, *i.e.* 1004 instances of non-difficult objects. The annotated views for cars are distributed quite homogeneously across all the poses, although they are slightly biased towards the frontal and rear views. Category *bus* provides only 301 samples, and the pose is clearly concentrated in the frontal view.

For the category *Car*, Figures 3.7(a) and 3.7(b) show that the performance of both models (continuous and discrete) are comparable. The continuous pose model tends to get confused with nearby views, while the discrete approach reports more errors with opposite viewpoints. The scenario changes when one inspects the results for the *Bus* object category. Figures 3.7(c) and 3.7(d) show that the performance of the continuous model is slightly worse than the one of the discrete model. Like we detail above, the continuous model tends to concentrate its predictions around the pose annotated bias (*i.e.* the frontal). Observe the bar diagram in 3.7(c), where most of the Rear views are assigned to Frontal views.

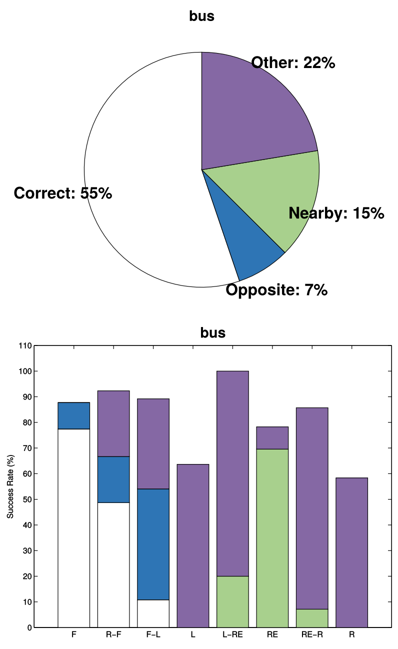
We want to conclude this analysis, adding an additional dimension to the discussion: the influence (in the performance) of the evaluation metric used. The problem of *simultaneous* detection and pose estimation has not been associated with either a clear experimental evaluation process or an evaluation metric. Obviously, part of the problem is that discrete and continuous approaches, being of a different nature, have been evaluated in



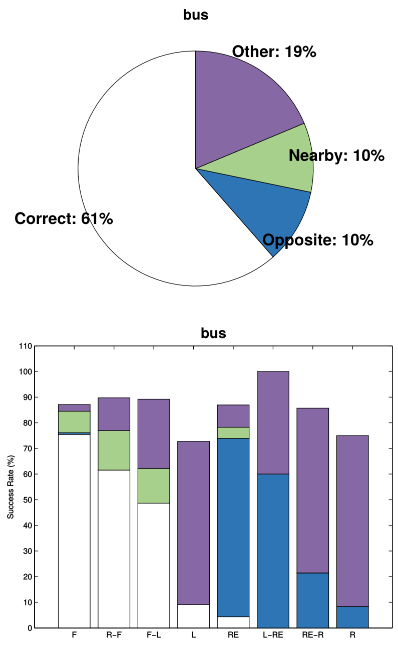
(a) Car - Euclidean loss.



(b) Car - Softmax loss.



(c) Bus - Euclidean loss.



(d) Bus - Softmax loss.

Figure 3.7: Viewpoint estimation performance detailed analysis. A comparison between continuous (with Euclidean loss) and discrete (with a Softmax loss) models for categories *Car* and *Bus*. (a) and (b) contain the results for the *car* category, while (c) and (d) show the results for the *bus* class. First row include pie charts showing the general performance of the models, where it is reported the percentage of: correct detections, confusions with opposite viewpoints, confusions with nearby poses, and the rest of errors (Other). Second row shows a detailed analysis, of the same type of errors, considering 8 set of viewpoints (F: Frontal, R-F: Right-Frontal, F-L: Frontal-Left, RE: Rear, RE-R: Rear-Right, L: Left, L-RE: Left-Rear and R: Right).

different ways. As a result, multiple evaluation metrics have been proposed, *e.g.* Pose Estimation Average Precision (PEAP) (Lopez-Sastre et al. 2011), Average Orientation Similarity (AOS) (Geiger et al. 2012) and AVP (Xiang et al. 2014). We refer the reader to (Redondo-Cabrera et al. 2016), where an extensive analysis of the different evaluation metrics is presented.

We have compared the performance of the AVP and the AOS metrics. Our experiments reveal that the AVP metric tends to favor discrete approaches, while the AOS metric favors the continuous models. For instance, for the category *bus*, Figure 3.8 shows the precision-recall curves when the different metrics are used. When the AVP metric is used, the discrete approach (Dis-AVP) obtains a higher average precision, compared to the one reported for the continuous model (Cont-AVP). On the other hand, when the AOS metric is followed, the average precision is slightly superior for the continuous model, *i.e.* Dis-AOS $<$ Cont-AOS.

In any case, taking into account the observations made in the work of Redondo-Cabrera et al. (2016), we would like to remark that the AOS metric is not an adequate measurement of the object detection and pose estimation problem. Redondo-Cabrera et al. (2016) show that this metric is dominated mainly by the detection performance, masquerading the pose estimation precision. Therefore, for the rest of our study, we choose to use an evaluation procedure based only on the AVP metric.

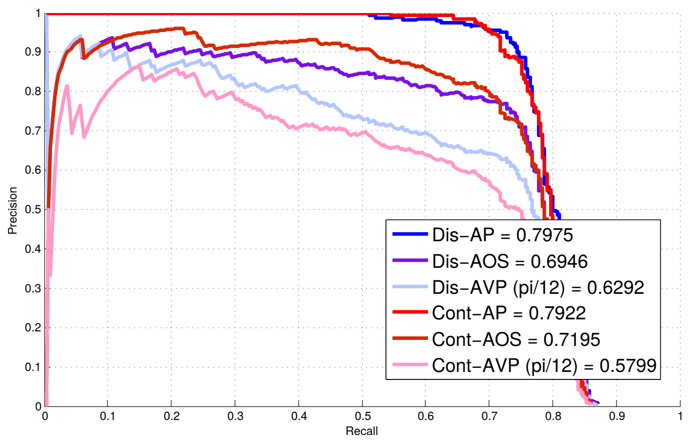


Figure 3.8: Detection and pose estimation performance for the *bus* category. A comparison based on evaluation metrics AOS and AVP, for both continuous (red tonalities) and discrete (blue tonalities) approaches.

Overall, based on these results, we conclude that continuous viewpoint estimation models tend to accumulate errors at nearby poses, while discrete pose estimation approaches errors are more likely to occur in opposite views.

Objectively, errors with close poses are not as important as errors associated with opposite poses. We believe that the continuous models could result in more attractive for the problem we are dealing with. However, if the amount of training data is not large enough, and is not well balanced in terms of pose annotations, a discrete estimation model,*i.e.* based on a classifier, is the best option. This is the normal situation in all datasets, and also in the PASCAL3D+. Therefore, *for the rest of our study*, we opt for a discrete model.

3.4.2.3 Independent vs. Joint object detection and pose estimation

A quick reading of the scientific literature reveals two main models for tackling the problem of detecting and estimating the pose of object categories. On the one hand, we find those who decouple both tasks. The detector is trained and executed separately to locate objects in the images. Subsequently, the pose estimator is responsible for associating a pose to the detected object. On the other hand, we have models that are trained to solve both tasks together. In this section, we analyze the performance of these two families of works. To do so, we offer a detailed comparison of the proposed architectures in Section 3.3.2, with existing state-of-the-art models that belong to one family or another.

We need to start this experimental evaluation making the following observations with respect to the three architectures proposed in this paper. Technically, our 3 network designs present a clear evolution in terms of the degree of coupling of the tasks of detection and pose estimation. Our *Single-path* approach clearly belongs to the *joint* family. Note that in this architecture, all the features of the network are shared for both tasks. With the *Specific-path* architecture we advance one step forward in the decoupling degree. It is a *hybrid* system, where the convolutional layer features are shared, while the FC layers are split into two paths: one for the object localization and one for the pose estimation. Finally, we propose the *Specific-network*. Although it should be considered as an architecture belonging to the group of *independent*, we cannot forget that it actually proposes a new paradigm, where both networks, specialized in different tasks, can be trained end-to-end. Note that although the networks learn their characteristics in a decoupled way, the ROIs produced by the network in charge of the location are shared with the network for the estimation of the pose, which somehow conditions their learning. This end-to-end methodology clearly differs from the rest of state-of-the-art *independent* models (*e.g.* (Tulsiani and Malik 2015)).

Table 3.3 shows the results for all of our architectures in the PASCAL3D+

Methods	Aero	Bike	Boat	Bus	Car	Chair	Table	MBike	Sofa	Train	Monitor	Avg.
AP Object Detection												
Single-path	78.1	**74.3**	47.2	**79.7**	70.2	28.2	**53.0**	76.0	52.0	**79.5**	60.8	63.6
Specific-path	**78.5**	73.1	**49.3**	79.2	**70.3**	**32.3**	52.7	78.0	**58.0**	77.9	**64.6**	**64.9**
Specific-network	77.8	74.2	47.9	78.7	**70.3**	30.7	52.9	**78.1**	56.5	77.7	62.7	64.3
AVP 4 Views - Joint Object Detection and Pose Estimation												
Single-path	52.4	41.7	18.6	66.2	45.3	14.2	26.1	44.7	40.4	63.7	52.9	42.4
Specific-path	56.7	54.7	**24.1**	66.2	50.2	**17.3**	**30.1**	55.7	**44.0**	**61.6**	**60.4**	**47.4**
Specific-network	**58.4**	**57.0**	23.2	**66.3**	**53.3**	16.9	27.9	**60.9**	41.5	60.1	52.6	47.1
AVP 8 Views - Joint Object Detection and Pose Estimation												
Single-path	42.9	28.9	11.1	52.7	38.8	10.5	18.1	32.0	28.3	50.2	40.5	32.2
Specific-path	47.2	38.3	**16.3**	47.2	43.0	12.8	**25.5**	47.5	33.2	**53.4**	**43.5**	37.1
Specific-network	**51.3**	**43.2**	14.4	**54.6**	**46.1**	**13.3**	21.8	**48.4**	**33.8**	49.4	41.7	**38.2**
AVP 16 Views - Joint Object Detection and Pose Estimation												
Single-path	22.8	19.5	7.8	54.4	31.8	6.8	14.0	20.5	15.6	**42.9**	23.6	23.6
Specific-path	33.4	25.9	10.1	51.3	32.7	8.0	20.1	23.8	**25.9**	38.0	**32.5**	27.4
Specific-network	**36.7**	**30.5**	**11.7**	**57.4**	**39.7**	**8.9**	**21.8**	**29.6**	25.5	38.0	31.9	**30.2**
AVP 24 Views - Joint Object Detection and Pose Estimation												
Single-path	18.1	15.3	4.4	44.8	27.2	5.2	11.8	13.7	14.0	**36.9**	16.9	18.9
Specific-path	**26.0**	18.3	7.7	40.6	29.3	5.2	15.9	18.4	**20.3**	36.7	**24.4**	22.1
Specific-network	22.9	**21.8**	**8.8**	**45.0**	**33.2**	**7.0**	**18.2**	**20.8**	16.9	33.4	21.8	**22.7**

Table 3.3: Object detection and pose estimation results in the PASCAL3D+ dataset. Comparison between all our architectures. In gray color we show our *joint* solution, *i.e.* the *Single-path* architecture. The remaining architectures (*Specific-path* and *Specific-network*) can be classified in the group of *independent* approaches.

dataset. Overall, our two *independent* models report a better performance than the *Single-path* architecture. For the specific case of 4 set of views, the best performance is given by the *Specific-path* model, which achieves the best AVP for 6 of 11 categories. For the rest of the set of views (8, 16 and 24), the best performance is consistently achieved by our *Specific-network* architecture. The obtained results show that the *independent* approaches perform better than *joint* approaches. In Figure 3.9, we show some qualitative examples produced by our *Specific-network* architecture.

We now compare our best model, *i.e.* the *Specific-network*, with the state-of-the-art models in Table 3.4. First of all, our *Specific-network* reports the best object detection results: see last column in Table 3.4.

Depending on the number set of views used for the evaluation in the PASCAL3D+ we can identify different winners, even from different families of methods. For instance, *joint* models retrieve the best results, in terms of mAVP, for 4, 8 and 24 views. For 16 views, it is the *independent* model of Tulsiani and Malik (2015) the one reporting the best performance.

Regarding all the results in Table 3.4 we can conclude that the *independent* approaches exhibit a better accuracy over most of the *joint* models.

Note that the state of the art for 24 view sets is achieved by the Craft-

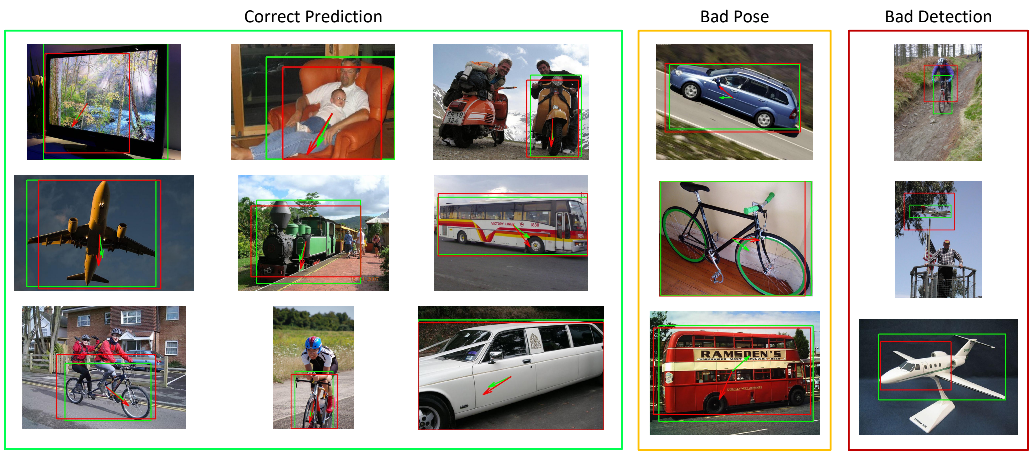


Figure 3.9: Qualitative results produced by the *Specific-network* in the PASCAL3D+ dataset. In green, we depict the ground truth annotations, while in red we show the results produced by our model. Rectangles correspond to the bounding boxes, while the arrows depict annotated orientations of the objects.

CNN (Massa et al. 2016), which uses synthetic CAD models during learning. This is also the case for the RenderCNN (Su et al. 2015). The rest of the models, including ours, do not use any extra data in form of CAD models. Note that the *Specific-network* systematically reports a better performance than the RenderCNN, for instance. The Single-Shot approach of Poirson et al. (2016) is the clear winner for 4 and 8 set of views, and the VP&KP (Tulsiani and Malik 2015) wins for 16 set of views. In all these scenarios, our *Specific-network* reports a higher detection accuracy than the winner model. This aspect is relevant because the metric used tends to favor detectors with a lower localization precision. We refer the reader to the study of Redondo-Cabrera et al. (2016) for more details. In other words, the more detections that are retrieved by a model, the greater the likelihood that the objects for which pose estimations have to be assigned are objects that, being more difficult to detect, appear occluded or truncated, or that are too small, aspects which naturally complicate a correct estimation of the viewpoint.

Every model comes with its own detector: VP&KP uses the R-CNN (Girshick et al. 2014), Craft-CNN uses the Fast R-CNN (Girshick 2015), and we follow the Faster R-CNN architecture (Ren et al. 2015). How can we evaluate the actual influence of the detector in the viewpoint estimation performance? In order to shed some light on this issue, we have decided to perform an additional experiment. We have taken the code of the VP&KP

Methods	Aero	Bike	Boat	Bus	Car	Chair	Table	MBike	Sofa	Train	Monitor	mAVP	mAP
AVP 4 Views - Joint Object Detection and Pose Estimation													
VDPM Xiang et al. (2014)	34.6	41.7	1.5	26.1	20.2	6.8	3.1	30.4	5.1	10.7	34.7	19.5	26.8
DPM-VOC+VP Pepik et al. (2012)	37.4	43.9	0.3	48.6	36.9	6.1	2.1	31.8	11.8	11.1	32.2	23.8	27.0
Craft-CNN Massa et al. (2016)	-	-	-	-	-	-	-	-	-	-	-	-	-
Single-Shot Poirson et al. (2016)	64.6	62.1	26.8-	70.0	51.4	11.3	40.7	62.7	40.6	65.9	61.3	**50.7**	61.0
SubCNN Xiang et al. (2017)	61.4	60.4	21.1	63.0	48.7	23.8	17.4	60.7	47.8	55.9	62.3	47.5	60.7
RenderCNN Su et al. (2015)	50.0	50.5	15.1	57.1	41.8	15.7	18.6	50.8	28.4	46.1	58.2	39.7	56.9
VP&KP Tulsiani and Malik (2015)	63.1	59.4	20.3	69.8	55.2	25.1	24.3	61.1	43.8	59.4	55.4	49.1	56.9
Specific-network	58.4	57.0	23.2	66.3	53.3	16.9	27.9	60.9	41.5	60.1	52.6	47.1	**64.3**
AVP 8 Views - Joint Object Detection and Pose Estimation													
VDPM Xiang et al. (2014)	23.4	36.5	1.0	35.5	23.5	5.8	3.6	25.1	12.5	10.9	27.4	18.7	29.9
DPM-VOC+VP Pepik et al. (2012)	28.6	40.3	0.2	38.0	36.6	9.4	2.6	32.0	11.0	9.8	28.6	21.5	28.3
Craft-CNN Massa et al. (2016)	-	-	-	-	-	-	-	-	-	-	-	-	-
Single-Shot Poirson et al. (2016)	58.7	56.4	19.9	62.4	42.2	10.6	34.7	58.6	38.8	61.2	49.7	**45.1**	60.4
SubCNN Xiang et al. (2017)	48.8	36.3	16.4	39.8	37.2	19.1	13.2	37.0	32.1	44.4	26.9	31.9	60.7
RenderCNN Su et al. (2015)	44.5	41.1	10.1	48.0	36.6	13.7	15.1	39.9	26.8	39.1	46.5	32.9	56.9
VP&KP Tulsiani and Malik (2015)	57.5	54.8	18.9	59.4	51.5	24.7	20.5	59.5	43.7	53.3	45.6	44.5	56.9
Specific-network	51.3	43.2	14.4	54.6	46.1	13.3	21.8	48.4	33.8	49.4	41.7	38.2	**64.3**
AVP 16 Views - Joint Object Detection and Pose Estimation													
VDPM Xiang et al. (2014)	15.4	18.4	0.5	46.9	18.1	6.0	2.2	16.1	10.0	22.1	16.3	15.6	30.0
DPM-VOC+VP Pepik et al. (2012)	15.9	22.9	0.3	49.0	29.6	6.1	2.3	16.7	7.1	20.2	19.9	17.3	28.3
Craft-CNN Massa et al. (2016)	-	-	-	-	-	-	-	-	-	-	-	-	-
Single-Shot Poirson et al. (2016)	46.1	39.6	13.6	56.0	36.8	6.4	23.5	41.8	27.0	38.8	36.4	33.3	60.0
SubCNN Xiang et al. (2017)	28.0	23.7	10.0	50.8	31.4	14.3	9.4	23.4	19.5	30.7	27.8	24.5	60.7
RenderCNN Su et al. (2015)	27.5	25.8	6.5	45.8	29.7	8.5	12.0	31.4	17.7	29.7	31.4	24.2	56.9
VP&KP Tulsiani and Malik (2015)	46.6	42.0	12.7	64.6	42.7	20.8	18.5	38.8	33.5	42.5	32.9	**36.0**	56.9
Specific-network	36.7	30.5	11.7	57.4	39.7	8.9	21.8	29.6	25.5	38.0	31.9	30.2	**64.3**
AVP 24 Views - Joint Object Detection and Pose Estimation													
VDPM Xiang et al. (2014)	8.0	14.3	0.3	39.2	13.7	4.4	3.6	10.1	8.2	20.0	11.2	12.1	29.5
DPM-VOC+VP Pepik et al. (2012)	9.7	16.7	2.2	42.1	24.6	4.2	2.1	10.5	4.1	20.7	12.9	13.6	27.1
Craft-CNN Massa et al. (2016)	42.4	37.0	18.0	59.6	43.3	7.6	25.1	39.3	29.4	48.1	28.4	**34.4**	59.9
Single-Shot Poirson et al. (2016)	33.4	29.4	9.2	54.7	35.7	5.5	23.0	30.3	27.6	44.1	34.3	28.8	59.3
SubCNN Xiang et al. (2017)	20.7	16.4	7.9	34.6	24.6	9.4	7.6	19.9	20.0	32.7	18.2	19.3	60.7
RenderCNN Su et al. (2015)	21.5	22.0	4.1	38.6	25.5	7.4	11.0	24.4	15.0	28.0	19.8	19.8	56.9
VP&KP Tulsiani and Malik (2015)	37.0	33.4	10.0	54.1	40.0	17.5	19.9	34.3	28.9	43.9	22.7	31.1	56.9
Specific-network	22.9	21.8	8.8	45.0	33.2	7.0	18.2	20.8	16.9	33.4	21.8	22.7	**64.3**

Table 3.4: Comparison with the state-of-the-art in the PASCAL3D+ dataset.In gray color we show the *joint* solutions.

model provided by the authors. This model defines an *independent* type architecture, where two completely decoupled and different deep networks are used: one for detection, and one for the pose estimation. We start using our *Specific-path* model which has the best detection performance, and we run it over the training images. We then collect these detections on the training data to enrich the ground truth data. Note that we only collect those detections whose overlap with the original ground truth is greater than 70%. This is equivalent to the jittering technique applied in the original paper but taking into account the bounding box distribution of the detector. With this *extended* training data, we proceed to train the original pose estimator of Tulsiani and Malik (2015). For the test images, we recover our detections and apply the described pose estimator on them. We call this pipeline: Improved VP&KP (Imp-VP&KP). Technically, the detector of the original VP&KP has been improved, using the Faster R-CNN now.

As we can see in Table 3.5, our Improved VP&KP systematically reports better results than the original work. Moreover, in Figure 3.10 we present

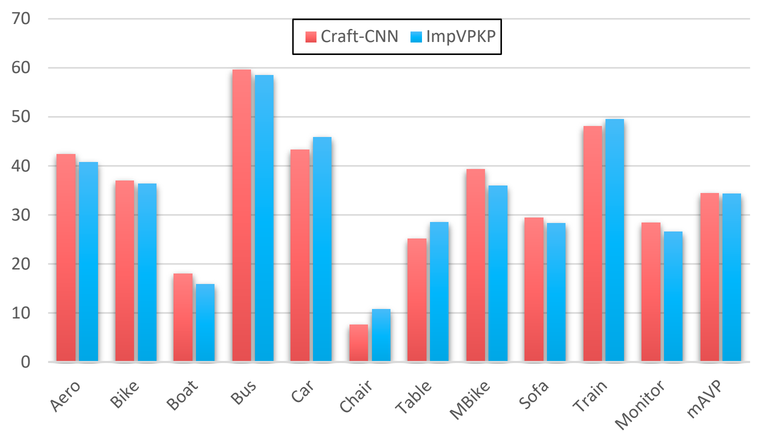


Figure 3.10: Comparison of Craft-CNN and the Imp-VP&KP experiment for 24 views.

Methods	Aero	Bike	Boat	Bus	Car	Chair	Table	MBike	Sofa	Train	Monitor	Avg.
AVP 4 Views - Joint Object Detection and Pose Estimation												
VP&KP Tulsiani and Malik (2015)	63.1	59.4	20.3	69.8	55.2	25.1	24.3	61.1	43.8	59.4	55.4	49.1
Imp-VP&KP	70.8	66.2	37.9	75.5	61.6	17.7	39.5	68.9	49.6	67.0	62.8	56.1
AVP 8 Views - Joint Object Detection and Pose Estimation												
VP&KP Tulsiani and Malik (2015)	57.5	54.8	18.9	59.4	51.5	24.7	20.5	59.5	43.7	53.3	45.6	44.5
Imp-VP&KP	63.9	61.4	29.0	63.3	56.2	15.8	32.8	65.3	42.0	60.6	53.6	49.4
AVP 16 Views - Joint Object Detection and Pose Estimation												
VP&KP Tulsiani and Malik (2015)	46.6	42.0	12.7	64.6	42.7	20.8	18.5	38.8	33.5	42.5	32.9	36.0
Imp-VP&KP	51.2	43.2	20.4	68.9	47.3	17.7	30.1	40.8	36.5	44.7	38.9	39.6
AVP 24 Views - Joint Object Detection and Pose Estimation												
VP&KP Tulsiani and Malik (2015)	37.0	33.4	10.0	54.1	40.0	17.5	19.9	34.3	28.9	43.9	22.7	31.1
Imp-VP&KP	40.7	36.4	15.8	58.5	45.8	10.7	28.5	35.9	28.3	49.5	26.6	34.3

Table 3.5: VP&KP (Tulsiani and Malik 2015) vs. Imp-VP&KP experiment.

a comparison between the Imp-VP&KP and the results of the Craft-CNN (Massa et al. 2016) for 24 views. We can observe how by simply updating the object detector, the model of Tulsiani and Malik (2015) can easily get the same performance as the Craft-CNN (Massa et al. 2016).

3.4.2.4 The side effect of the pose estimation in the joint system

The systems that address the object detection and pose estimation problems simultaneously, in principle, have multiple benefits, compared with the models that decouple both tasks. They are clearly more efficient, in terms of computational cost. Note that during training, for instance, both tasks are learned simultaneously. Moreover, for a test image, the localization of the object, and the estimation of its pose is obtained at the same time, not needing to process the images with a complex pipeline consisting of a detector followed by a viewpoint estimator. In a joint system, most of the operations are shared between tasks.

In spite of these advantages, our experiments reveal that there is a trade-off between doing the object localization accurately and casting a precise estimation for the viewpoint. Ideally, a good detector should be invariant to the different poses of an object, *e.g.* it should correctly localize frontal and rear views of cars. This would push the detection models to learn representations that are not adequate to discriminate between the different poses, being this what a good pose estimator should learn.

In Table 3.6 we report some results that can help us to understand the mentioned trade-off. The first two rows show the results reported by Massa et al. (2016). They show a comparison between their joint and independent approaches. Their independent solution clearly obtains a better performance for the object detection than the joint model, but also one can observe how the pose estimation precision, in terms of mAVP, decreases.

It is also interesting to observe, in the last rows of Table 3.6, how this trade-off between object detection and pose estimation performances also affects the model of Poirson et al. (2016). We can see that when they try to train their Single-Shot joint model to be more discriminative in terms of poses,*i.e.* increasing the number of sets of views from 4 to 24, the object detection accuracy tends to decrease.

If we now analyze the performance reported by our solutions, from the *Single-path* to the *Specific-network*, we note that the detection performance slightly increases for our *independent* models, but we are able to also report better performance in terms of pose estimation. We explain this fact with the type of deep architectures we have proposed. Both the *Specific-path* and the *Specific-network* cannot be categorized as truly independent models: we do not completely decouple the tasks of object localization and pose estimation. Ours is an exercise or relaxing the amount of shared information between these tasks, which defines a training process able to enforce the networks to learn representations that are adequate for both tasks.

3.4.3 Results in the ObjectNet3D dataset

In this work, we also perform a detailed experimental evaluation of our models in the large scale dataset for 3D object recognition ObjectNet3D developed by Xiang et al. (2016). It consists of 100 categories, 90.127 images, and more than 200.000 annotated objects. This dataset has been carefully designed for the evaluation of the problems of object detection, classification, and pose estimation. Similarly to the PASCAL3D+, the object pose annotation is the result of the manual alignment of a 3D CAD model with the target object.

Method	mAP	mAVP
Craft-CNN (AlexNet) Massa et al. (2016)		
Joint - 24 views	48.6	21.1
Independent - 24 views	51.6	20.5
Ours		
Joint - 24 views (Single-path)	63.6	18.9
Independent - 24 views (Specific-path)	64.9	22.1
Independent - 24 views (Specific-network)	64.3	22.7
Single-Shot Poirson et al. (2016)		
Joint - 4 views	61.0	50.7
Joint - 8 views	60.4	45.1
Joint - 16 views	60.0	33.3
Joint - 24 views	59.3	28.8

Table 3.6: Analysis of the trade-off between object detection and pose estimation performance.

Figure 3.11 shows some examples of this dataset.

Like we describe in Section 3.4.1, we strictly follow the experimental setup detailed in (Xiang et al. 2016). Only the training data is used to learn the models. We then report our results using the validation and test sets. For the evaluation metric, Xiang et al. (2016) propose a generalization of the AVP. They basically extend the AVP to consider the prediction of the three angles provided in the annotation: azimuth, elevation, and in-plane rotation. Technically, the solutions must provide an estimation for these three variables. Then, the corresponding predicted rotation matrix $\hat{R}$ is constructed. The difference between the ground truth pose R, and the prediction encoded in $\hat{R}$ is computed using a geodesic distance as follows:

$$d(R, \hat{R}) = \frac{1}{\sqrt{2}} || \log \left(\hat{R}^T R \right) || \,. \tag{3.8}$$

According to Xiang et al. (2016), for the AVP, an estimation is considered to be correct if $d(R, \hat{R}) < \frac{\pi}{6}$.

With respect to the technical implementation of our models, note that they cast a prediction for the three pose angles (azimuth, elevation and in-plane rotation) simultaneously. We repeat the same initialization procedure, using a pre-trained model on the ImageNet dataset. Again, we use the Stochastic Gradient Descent optimizer, with a momentum of 0.9, and the weight decay is set to 0.0005. This time we fix to 1 all the specific learning rates for each layer. The training is performed in an end-to-end fashion

Figure 3.11: ObjectNet3D image samples.

Losses	mAP	mAVP
Discrete (Eq. 3.4)	59.7	40.9
Euclidean (Eq. 3.5)	60.5	41.2
Huber (Eq. 3.6)	60.4	41.5

Table 3.7: Loss function analysis for the ObjectNet3D dataset. Object detection and viewpoint estimation performances are reported.

following the Faster R-CNN procedure (Ren et al. 2015).

3.4.3.1 Discrete vs. Continuous approaches analysis

In our experiments with the PASCAL3D+ dataset, one of the main conclusions obtained has been that the discrete pose estimation models, based on classifiers, give better results than continuous pose estimation models. When the number of training samples is not large enough, and the pose annotations are not well balanced, a discrete estimation model is generally the best option. Now, with the novel ObjectNet3D dataset, which provides more viewpoint annotations for more object categories, we have the opportunity to explore whether we can obtain better performance for the continuous approaches.

We follow the same procedure described in Section 3.4.2.2 for our previous Discrete vs. Continuous approaches analysis. We use our *Single-path* model, which is trained for a continuous pose estimation task, solving a regression problem using the Euclidean and Huber losses. When the discrete pose estimation problem is tackled, we simply learn a classifier employing the Softmax loss.

Table 3.7 reports the obtained results of our *Single-path* architecture, trained on the training set, and evaluated over the validation set. In our experiments, we observe a similar performance among all the models, but this time the continuous pose estimation architectures exhibit a small advantage as we have previously suggested. Therefore, for the rest of the experiments in this dataset, we use the continuous viewpoint architecture, employing the Huber loss.

3.4.3.2 Comparison of our architectures

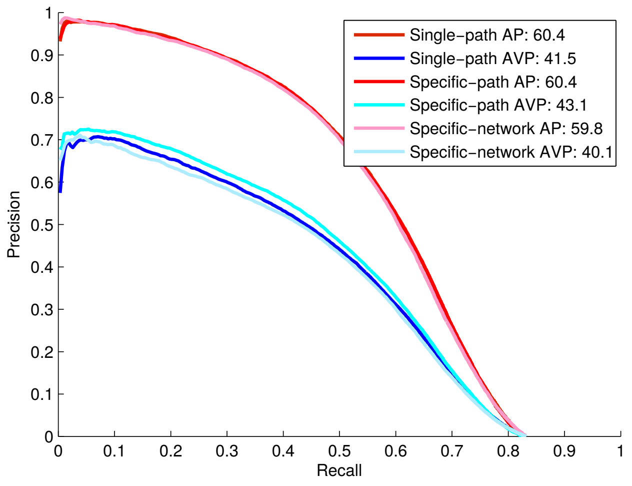


Figure 3.12: Object detection and pose estimation performance of our Single-path, Specific-path and Specific-network architectures in the ObjectNet3D dataset. Both AP and AVP metrics, with their associated precision-recall curves, are reported.

In this section, we propose to analyze the performance of all our architectures, *i.e.* the Single-path, the Specific-path, and the Specific-network, in this novel dataset. We only use the training set for learning the models, and the evaluation is carried in the validation set. Figure 3.12 shows that for this dataset, all our models report a very similar performance. Note how the AP reported for the object localization task is almost identical for the three networks, while for the pose estimation the Specific-path exhibits a slightly superior AVP. In any case, we conclude that for this dataset, there is no clear winner within our models. Therefore, now that the amount of training data in the ObjectNet3D dataset has increased considerably, it seems that there are no major differences between treating the problem of locating objects and estimating their pose jointly or separately.

Method	mAP	mAVP
AlexNet Xiang et al. (2016)	54.2	35.4
VGG-16 Xiang et al. (2016)	**67.5**	42.6
Our	64.2	**46.7**

Table 3.8: Comparison with state-of-the-art models in the ObjectNet3D dataset.

3.4.3.3 A comparison with the state of the art

In this section, we provide a comparison with the state-of-the-art models reported by Xiang et al. (2016). For the joint 2D detection and continuous 3D pose estimation task, they propose a modification of the Fast R-CNN (Girshick 2015) model, using two different base architectures: the VGG16 (Simonyan and Zisserman 2014) and the AlexNet (Krizhevsky et al. 2012). Technically, they add a viewpoint regression FC branch just after the FC7 layer. Their network is trained to jointly solve three tasks: classification, bounding box regression, and viewpoint regression. The FC layer for viewpoint regression is of size 3×101, *i.e.* , for each class, it predicts the three angles of azimuth, elevation and in-plane rotation. The smoothed L1 loss is used for viewpoint regression.

Table 3.8 shows the comparison with the state-of-the-art models, but now in the test set of the ObjectNet3D dataset. On the first two rows of the table, we include the results of the VGG-16 and the AlexNet based models reported by Xiang et al. (2014). The last row shows the performance of our *Specific-path* model. First, note that we are able to report a better detection performance than the AlexNet based solution of Xiang et al. (2014). Second, although the VGG16 based architecture of Xiang et al. (2014) reports the best detection results, if we simultaneously consider the object localization and viewpoint estimation accuracies, *i.e.* using the AVP metric, our model is the clear winner. This fact is particularly relevant if we consider that the pose estimation is bounded by the detection performance, according to the evaluation metric used. Overall, this implies that our model is more accurate in predicting poses.

In a detailed comparison of our solution with the VGG16 based architecture used by Xiang et al. (2014), we find the following differences that also help to explain the results obtained. First, while our model is trained fully end-to-end, the approach of Xiang et al. (2016) consists in training the Region Proposal Network of Ren et al. (2015) first, and then using these proposals to fine-tune their model for the object detection and pose estimation tasks. Therefore, their model is mainly trained to optimize the detection per-

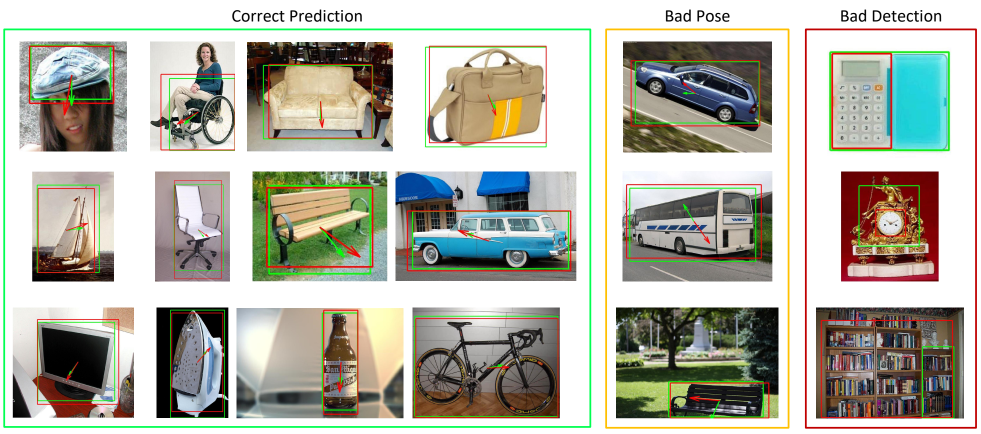


Figure 3.13: Qualitative results produced by the Specific-path in the ObjectNet3D. In green, we depict the ground truth annotations, while in red we show the results produced by our model. Rectangles correspond to the bounding boxes, while the arrows depict annotated orientations of the objects.

formance, which explains why our *Specific-path* reports a slightly lower mAP. Second, there are significant differences in how the pose estimation is performed. In the work of Xiang et al. (2016), a regressor is trained to directly predict viewpoint values in degrees. We, instead, decompose each angle into two polar coordinates. This decomposition naturally takes into account the cyclic nature of viewpoint angles. This explains why our *Single-path* model reports a better performance for the pose.

We finally show some qualitative results for the ObjectNet3D dataset in Figure 3.13.

3.5 Conclusion

This chapter has contributed to the research community with a complete analysis of the state of the art for the problem of simultaneous object detection and pose estimation. We have designed an experimental validation, using the PASCAL3D+ and ObjectNet3D datasets, where we can evaluate the degree of coupling that exists among the tasks of object localization and viewpoint estimation. For doing so, we have introduced three deep learning architectures, which are able to perform a joint detection and pose estimation, where we gradually decouple these two tasks. With the proposed models we have achieved the state-of-the-art performance in both datasets. We have

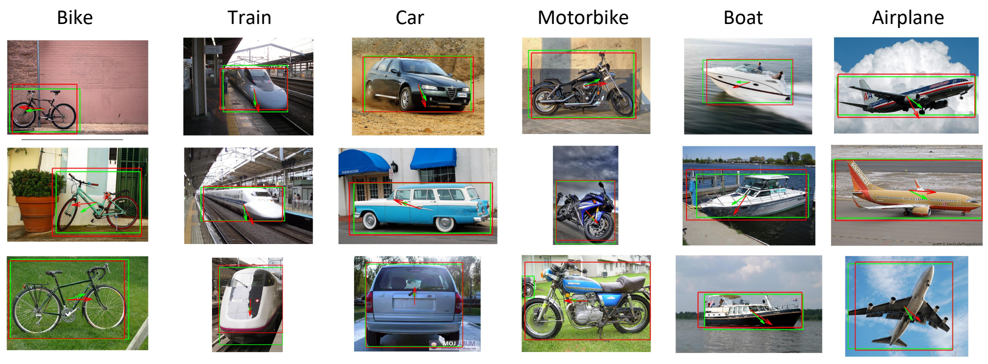


Figure 3.14: Object detection and pose estimation qualitative examples produced by our model for some of the most common vehicles types. In green is depicted the annotated ground truth for the bounding box and the pose; in red are drawn the predicted bounding box and the estimated pose.

concluded that decoupling the detection from the viewpoint estimation task have benefits on the overall performance of the models.

Furthermore, we have extended the comparative analysis of all our approaches considering the pose estimation as a discrete or a continuous problem, according to the two families of works in the literature. In our experiments, we have analyzed the main factors that need to be considered during the system design and training. Despite the similar performance among the different approaches, we have observed a difference between the discrete and the continuous models. We conclude that the continuous approaches are more sensitive to the pose bias in the annotation than the discrete models.

How far are we from a realistic ITS application? The models developed in this work achieve good performance and they could potentially be applied to ITS application. In Figure 3.14 we show some extra qualitative examples produced by our best model for a broad type of vehicles such as ground, rail, naval, or aerial. Despite the good performance in detecting objects, the pose estimation is still far from a robust solution. The reason is given by the difficulty of the problem, due to two factors:

1. The bias in the dataset annotation and construction. The data collection process of natural images often leads a bias for the most common views of the objects.

2. The similarity in terms of appearance between different poses. In special, for those objects with a high degree of symmetry.

The first step to fight against these difficulties and towards a realistic ITS solutions, consists in expanding the existing datasets. Increasing the diversity of the collected samples, and having uniform collections of viewpoints is an effective way towards the development of robust ITS applications.

Chapter 4

Contributions

We can only see a short distance ahead, but we can see plenty there that needs to be done.

`Alan Turing,` ***`Computing machinery and intelligence.`***

4.1 Summary and main contributions

The main goal of this thesis has been the design of new deep neural networks models to improve some of the existing techniques that are often used in some of the main problems of the ITS field. Specifically, this thesis has focused on two of the major problems in the ITS community: the vehicle counting problem and the simultaneous object detection and pose estimation. In the following lines the main contributions that are presented in this work are briefly summarized:

- The object counting problem has been addressed. During the development of this topic, two top-tier publications were obtained. The first publication was (Oñoro-Rubio and López-Sastre 2016). It was published in the European Computer Vision Conference (ECCV) in 2016 where 415 papers were accepted with an acceptance rate of 26.7%. This publication is cited by 121 scientific papers by the time this thesis is written, according to Google Scholar. The second publication is (Oñoro-Rubio et al. 2018). It was published in the British Machine Vision Conference (BMVC) in 2018, where 255 papers were published with an acceptance rate of 29.5%. In the following bullets the main scientific contributions are summarized:

  - An extensive overview of the most commonly used datasets and thorough review of the literature have been proposed, focusing on the object counting problem.
  - The problem of the scale associated with the perspective of the scene has been covered. In previous works, such as in (Chan et al. 2008), this problem has been addressed by adding a perspective map that was used to scale the features. This thesis has studied the impact of the perspective map on the performance. In addition, two new neural network architectures were proposed: the CCNN and the Hydra model. The CCNN is a single scale fully convolutional neural network that learns to map image patches into density maps. The Hydra model is a perspective-aware neural network that learns an internal representation that combines multiple scales at the same time. The proposed ideas have been exhaustively evaluated on three of the most popular datasets.
  - With the effectiveness of the skip-connection mechanism for the training of deep neural networks, and with the succeed of the hourglass networks in the field, a new self-gating mechanism which is applied to the short-cut connections of the U-Net was introduced. The proposed mechanism helps to block the potentially noisy features that are passed to the top layers by learning to control the information flow that is forwarded in the short-cut connections. A complete set of experiments has shed light on the working mechanism of the proposed idea, and prove the effectiveness of the proposed mechanism by testing it in several datasets.
  - A complete review of the most recent state-of-the-art methods and a complete discussion were proposed.

- The object detection and pose estimation topic have been covered during the second half of this thesis. As for the result, the article (Oñoro-Rubio et al. 2018) was published in the Image and Vision Computing journal with a 2.875 5-year impact factor and Q1 quartile. In the lines below the contributions introduced by this thesis in the object detection and pose estimation field are summarized:

  - An extensive review of the problem with a complete summary of the most relevant works and methods have been presented.
  - The compromise between object detection and the viewpoint estimation tasks have been spotted. Detecting objects requires representations that are invariant to the viewpoint while estimating

the viewpoint needs to be a discriminative task. With that in mind, the Faster-RCNN was modified leading to three novel deep learning architectures where the object detection and the viewpoint estimation tasks were gradually decoupled. A complete set of experiments on two important datasets made evident the aforementioned trade-off. With our experiments, we observed how the proposed decoupled models such as the Specific-network or the Specific-path, perform better than our Single-path model, which fully couples both tasks.

– The problem of whether the pose estimation should be solved as a classification problem, or as a regression problem has been also explored. Despite a similar performance among all the approaches, it has been observed small differences which showed that the continuous models are more sensitive to the pose bias of the annotation than the discrete approaches.

4.2 Future research

There is still much work to be done in the problems addressed in this thesis before the solutions can be efficiently implemented in a realistic ITS product. Despite the extremely fast development of the AI field impulsed by many companies and governments, there is still a gap between the developed models in the laboratory and their deployment in a real-world scenario. Although this thesis has presented several models that have been shown to be powerful for the object counting and for the object detection and pose estimation problems, these are the main directions for future work:

- The current counting by regression systems tend to underestimate the total count. It seems there is a bias on the systems where the estimated amount of objects is on average inferior to the ground truth. This effect is probably caused by the density maps that are used as background. The density maps tend to have large areas with 0 values which can be the reason of the undercounting effect. Further research and new systems are needed, where alternatives to the density map estimation are evaluated.

- In this thesis, the object detection and pose estimation problem has been deeply analyzed. However, extracting more explanations of the predictions for the pose may be helpful to understand what are the main difficulties of the current solutions. Extending some recent models, like

(Ribeiro et al. 2016), and (Selvaraju et al. 2017), could allow us to discover new difficulties that need to be addressed.

- The problem of domain adaptation is an active research area (Ganin and Lempitsky 2015), and (Redko et al. 2017). The performance of a system trained for a certain dataset can drastically be reduced when it is tested on another dataset. Therefore, extending the proposed models for efficiently adapting the trained models to new scenarios without adding additional labeled data to the training set is a fundamental problem.

- Data acquisition and data labeling are the utmost fundamental steps that directly impact on system performance ((Uijlings et al. 2018), and (Konyushkova et al. 2018)). However, this is an expensive process that involves human supervision. Therefore, to explore weakly-supervised or unsupervised approaches for the problems proposed could be an interesting direction for future work.

- Out of distribution situation is a problem of any AI system. In a deployed system, it may happen that the test distribution changes over time. Having a mechanism to detect when the test distribution has changed with respect to the trained model is an open and active research line (DeVries and Taylor 2018). An evaluation of the impact of this type of technique in the final performance of the solutions is a direction worth to be explored.

Bibliography

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `https://www.tensorflow.org/`. Software available from tensorflow.org.

C. Arteta, V. Lempitsky, J. A. Noble, and A. Zisserman. Interactive object counting. In *ECCV*, 2014.

Deepak Babu Sam, Shiv Surya, and R. Venkatesh Babu. Switching convolutional neural network for crowd counting. In *CVPR*, July 2017.

Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *ECCV*, pages 404–417, 2006.

Lucas Beyer, Alexander Hermans, and Bastian Leibe. Biternion nets: Continuous head pose regression from discrete training labels. In *Pattern Recognition*, volume 9358 of *Lecture Notes in Computer Science*, pages 157–168. Springer, 2015.

Gabriel J. Brostow and Roberto Cipolla. Unsupervised bayesian detection of independent motion in crowds. In *CVPR*, 2006.

Xinkun Cao, Zhipeng Wang, Yanyun Zhao, and Fei Su. Scale aggregation network for accurate and efficient crowd counting. In *ECCV*, 2018.

C. Caraffi, T. Vojir, J. Trefny, J. Sochman, and J. Matas. A system for real-time detection and tracking of vehicles from a single car-mounted camera. In *ITS Conference*, 2012.

A.-B. Chan, Z.-S.-J. Liang, and N. Vasconcelos. Privacy preserving crowd monitoring: Counting people without people models or tracking. In *CVPR*, 2008.

Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

Terrance DeVries and Graham W Taylor. Learning confidence for out-of-distribution detection in neural networks. *arXiv preprint arXiv:1802.04865*, 2018.

Mohamed Elhoseiny, Tarek El-Gaaly, Amr Bakry, and Ahmed M. Elgammal. A comparative analysis and study of multiview cnn models for joint object categorization and pose estimation. In *ICML*, 2016.

M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (VOC) challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010.

Pedro F. Felzenszwalb, Ross B. Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2010.

M. Fenzi, L. Leal-Taixé, B. Rosenhahn, and J. Ostermann. Class generative models based on feature regression for pose estimation of object categories. In *CVPR*, 2013.

Luca Fiaschi, Ullrich KÃűthe, Rahul Nair, and Fred A. Hamprecht. Learning to count with regression forest and structured labels. In *ICPR*, 2012.

Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *ICML*, 2015.

A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the KITTI vision benchmark suite. In *CVPR*, 2012.

Ross Girshick. Fast R-CNN. In *ICCV*, 2015.

Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

D. Glasner, M. Galun, S. Alpert, R. Basri, and G. Shakhnarovich. Viewpoint-aware object detection and continuous pose estimation. *Image and Vision Computing*, 30(12):923–933, 2012.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, 2010.

R. Guerrero-Gomez-Olmedo, R. J. Lopez-Sastre, S. Maldonado-Bascon, and A. Fernandez-Caballero. Vehicle tracking by simultaneous detection and viewpoint estimation. In *IWINAC*, 2013.

R. Guerrero-Gómez-Olmedo, B. Torre-Jiménez, R. López-Sastre, S. Maldonado-Bascón, and D. Oñoro Rubio. Extremely overlapping vehicle counting. In *Iberian Conference on Pattern Recognition and Image Analysis (IbPRIA)*, 2015.

Ricardo Guerrero-Gómez-Olmedo, Roberto J. López-Sastre, Saturnino Maldonado-Bascón, and Antonio Fernández-Caballero. Vehicle tracking by simultaneous detection and viewpoint estimation. In *Natural and Artificial Computation in Engineering and Medical Applications*. Springer Berlin Heidelberg, 2013.

C. Harris and C. Stennett. Rapid - a video rate object tracker. In *BMVC*, 1990.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

Haroon Idrees, Imran Saleemi, Cody Seibert, and Mubarak Shah. Multi-source multi-scale counting in extremely dense crowd images. In *CVPR*, 2013.

Haroon Idrees, Muhmmad Tayyab, Kishan Athrey, Dong Zhang, Somaya Al-Maadeed, Nasir Rajpoot, and Mubarak Shah. Composition loss for counting, density map estimation and localization in dense crowds. In *ECCV*, 2018.

Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014a.

Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014b.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

Ksenia Konyushkova, Jasper Uijlings, Christoph H. Lampert, and Vittorio Ferrari. Learning intelligent dialogs for bounding box annotation. In *CVPR*, 2018.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

Issam H. Laradji, Negar Rostamzadeh, Pedro O. Pinheiro, David Vazquez, and Mark Schmidt. Where are the blobs: Counting by localization with point supervision. In *ECCV*, 2018.

Yann LeCun, Bernhard E. Boser, John S. Denker, Donnie Henderson, R. E. Howard, Wayne E. Hubbard, and Lawrence D. Jackel. Handwritten digit recognition with a back-propagation network. In *NIPS*, pages 396–404, 1990.

Bastian Leibe, Edgar Seemann, and Bernt Schiele. Pedestrian detection in crowded scenes. In *CVPR*, 2005.

V. Lempitsky and A. Zisserman. Learning to count objects in images. In *NIPS*, 2010.

Guanbin Li and Yizhou Yu. Visual saliency based on multiscale deep features. In *CVPR*, 2015.

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 *ICPR*, 2009.

Min Li, Zhaoxiang Zhang, Kaiqi Huang, and Tieniu Tan. Estimating the number of people in crowded scenes by mid based foreground segmentation and head-shoulder detection. In *ICPR*, 2008.

Yuhong Li, Xiaofan Zhang, and Deming Chen. Csrnet: Dilated convolutional neural networks for understanding the highly congested scenes. In *CVPR*, 2018.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common objects in context. In *ECCV*, 2014.

Yun Liu, Krishna Kumar Gadepalli, Mohammad Norouzi, George Dahl, Timo Kohlberger, Subhashini Venugopalan, Aleksey S Boyko, Aleksei Timofeev, Philip Q Nelson, Greg Corrado, Jason Hipp, Lily Peng, and Martin Stumpe. Detecting cancer metastases on gigapixel pathology images. Technical report, arXiv, 2017. URL `https://arxiv.org/abs/1703.02442`.

Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2014.

Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, June 2015.

R. López-Sastre, C. Redondo-Cabrera, P. Gil-Jiménez, and S. Maldonado-Bascón. ICARO: Image collection of annotated real-world objects. http://agamenon.tsc.uah.es/Personales/rlopez/data/icaro, 2010.

R. J. Lopez-Sastre, T. Tuytelaars, and S. Savarese. Deformable part models revisited: A performance evaluation for object category pose estimation. In *ICCV 2011, 1st IEEE Workshop on Challenges and Opportunities in Robot Perception*, 2011.

D. G. Lowe. Object recognition from local scale-invariant features. In *CVPR*, volume 2, pages 1150–1157 vol.2, Sept 1999.

David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, November 2004. ISSN 0920-5691.

Chen Change Loy, Ke Chen, Shaogang Gong, Tao Xiang, Ke Chen, Shaogang Gong, Tao Xiang, Chen Change Loy, Ke Chen, Shaogang Gong, and Tao Xiang. Crowd counting and profiling: Methodology and evaluation, 2013.

Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *IJCAI*, 1981.

Andrew L. Maas, Awni Y. Hannun, and Andrew Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *in ICML Workshop on Deep Learning for Audio, Speech and Language Processing*, 2013.

Francisco Massa, Renaud Marlet, and Mathieu Aubry. Crafting a multi-task CNN for viewpoint estimation. In *BMVC*, 2016.

Daniel Oñoro-Rubio and Roberto J. López-Sastre. Towards perspective-free object counting with deep learning. In *ECCV*, 2016.

Daniel Oñoro-Rubio, Roberto Javier López-Sastre, Carolina Redondo-Cabrera, and Pedro Gil-Jiménez. The challenge of simultaneous object detection and pose estimation: A comparative study. *Image and Vision Computing*, 79:109–122, 2018.

Daniel Oñoro-Rubio, Niepert Mathias, and Roberto J. López-Sastre. Learning short-cut connections for object counting. In *BMVC*, 2018.

M. Ozuysal, V. Lepetit, and P. Fua. Pose estimation for category specific multiview object localization. In *CVPR*, 2009.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *ICML*, 2013.

Bojan Pepik, Michael Stark, Peter Gehler, and Bernt Schiele. Teaching 3D geometry to deformable part models. In *CVPR*, June 2012.

Viet-Quoc Pham, Tatsuo Kozakaya, Osamu Yamaguchi, and Ryuzo Okada. Count forest: Co-voting uncertain number of targets using random forest for crowd density estimation. In *ICCV*, December 2015.

Patrick Poirson, Phil Ammirato, Cheng-Yang Fu, Wei Liu, Jana Kosecka, and Alexander C. Berg. Fast single shot detection and pose estimation. In *3DV*, 2016.

Vincent Rabaud and Serge J. Belongie. Counting crowded moving objects. In *CVPR*, pages 705–711, 2006.

Viresh Ranjan, Hieu Le, and Minh Hoai. Iterative crowd counting. In *ECCV*, 2018.

Ievgen Redko, Amaury Habrard, and Marc Sebban. Theoretical analysis of domain adaptation with optimal transport. 2017.

J. Redmon and A. Farhadi. Yolo9000: Better, faster, stronger. In *CVPR*, 2017.

J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *CVPR*, 2016.

C. Redondo-Cabrera and R. Lopez-Sastre. Because better detections are still possible: Multi-aspect object detection with boosted hough forest. In *BMVC*, 2015.

C. Redondo-Cabrera, R. Lopez-Sastre, and T. Tuytelaars. All together now: Simultaneous object detection and continuous pose estimation using a hough forest with probabilistic locally enhanced voting. In *BMVC*, 2014.

C. Redondo-Cabrera, R. J. Lopez-Sastre, Y. Xiang, T. Tuytelaars, and S. Savarese. Pose estimation errors, the ultimate diagnosis. In *ECCV*, 2016.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NIPS*, 2015.

Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should I trust you?": Explaining the predictions of any classifier. In *ACM SIGKDD*, 2016.

Mikel Rodriguez, Ivan Laptev, Josef Sivic, and Jean-Yves Audibert. Density-aware person detection and tracking in crowds. In *ICCV*, 2011.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, volume 9351 of *Lecture Notes in Computer Science*, pages 234–241. Springer, 2015.

Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: an efficient alternative to sift or surf. In *ICCV*, 2011.

David Ryan, Simon Denman, Clinton Fookes, and Sridha Sridharan. Crowd counting using multiple local features. In *DICTA*, 2009.

L. Savarese, S.and Fei-Fei. View synthesis for recognizing unseen poses of object classes. In *ECCV*, 2008.

Cordelia Schmid. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006.

Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *ICCV*, 2017.

K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

Vishwanath A. Sindagi and Vishal M. Patel. Generating high-quality crowd density maps using contextual pyramid cnns. In *ICCV*, Oct 2017.

Amit Singhal. Modern information retrieval: a brief overview. *BULLETIN OF THE IEEE COMPUTER SOCIETY TECHNICAL COMMITTEE ON DATA ENGINEERING*, 24:2001, 2001.

Hao Su, Charles R. Qi, Yangyan Li, and Leonidas J. Guibas. Render for CNN: Viewpoint estimation in images using CNNs trained with rendered 3d model views. In *ICCV*, December 2015.

A. Thomas, V. Ferrari, B. Leibe, T. Tuytelaars, B. Schiele, and L. Van Gool. Towards multi-view object class detection. In *CVPR*, 2006.

Shubham Tulsiani and Jitendra Malik. Viewpoints and keypoints. In *CVPR*, 2015.

Shubham Tulsiani, João Carreira, and Jitendra Malik. Pose induction for novel object categories. In *ICCV*, 2015.

Oncel Tuzel, Fatih Porikli, and Peter Meer. Pedestrian detection via classification on riemannian manifolds. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(10):1713–1727, October 2008.

Jasper Uijlings, Stefan Popov, and Vittorio Ferrari. Revisiting knowledge transfer for training object class detectors. In *CVPR*, 2018.

J.R.R. Uijlings, K.E.A. van de Sande, T. Gevers, and A.W.M. Smeulders. Selective search for object recognition. *International Journal of Computer Vision*, 104(2):154–171, 2013.

Andreas Veit, Michael Wilber, and Serge Belongie. Residual networks behave like ensembles of relatively shallow networks. *Conference on Neural Information Processing Systems (NIPS)*, 2016.

Paul Viola and Michael J. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 2004.

Yu Xiang, Roozbeh Mottaghi, and Silvio Savarese. Beyond PASCAL: A benchmark for 3D object detection in the wild. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2014.

Yu Xiang, Wonhui Kim, Wei Chen, Jingwei Ji, Christopher Choy, Hao Su, Roozbeh Mottaghi, Leonidas Guibas, and Silvio Savarese. ObjectNet3D: A large scale database for 3D object recognition. In *ECCV*, 2016.

Yu Xiang, Wongun Choi, Yuanqing Lin, and Silvio Savarese. Subcategory-aware convolutional neural networks for object proposals and detection. In *IEEE Winter Conference on Applications of Computer Vision*, 2017.

S. Yingze-Bao, M. Sun, and S. Savarese. Toward coherent object detection and scene layout understanding. *Image and Vision Computing*, 29(9):569 – 579, 2011.

Cong Zhang, Hongsheng Li, Xiaogang Wang, and Xiaokang Yang. Cross-scene crowd counting via deep convolutional neural networks. In *CVPR*, June 2015.

Shanghang Zhang, Guanhang Wu, Joao P. Costeira, and Jose M. F. Moura. Understanding traffic density from large-scale web camera data. In *CVPR*, July 2017.

Yingying Zhang, Desen Zhou, Siqin Chen, Shenghua Gao, and Yi Ma. Single-image crowd counting via multi-column convolutional neural network. In *CVPR*, June 2016.

Those who can imagine anything, can create the impossible.
`Alan Turing.`