



Region-oriented Convolutional Networks for Object Retrieval

Bachelor's Thesis
Audiovisual Systems Engineering

Author: Eduard Fontdevila Bosch
Advisors: Xavier Giró-i-Nieto and Amaia Salvador Aguilera

Universitat Politècnica de Catalunya (UPC)
2014 - 2015

Abstract

This thesis is framed in the computer vision field, addressing a challenge related to instance search. Instance search consists in searching for occurrences of a certain visual instance on a large collection of visual content, and generating a ranked list of results sorted according to their relevance to a user query. This thesis builds up on existing work presented at the TRECVID Instance Search Task in 2014, and explores the use of local deep learning features extracted from object proposals. The performance of different deep learning architectures (at both global and local scales) is evaluated, and a thorough comparison of them is performed. Secondly, this thesis presents the guidelines to follow in order to fine-tune a convolutional neural network for tasks such as image classification, object detection and semantic segmentation. It does so with the final purpose of fine tuning SDS, a CNN trained for both object detection and semantic segmentation, with the recently released Microsoft COCO dataset.

Keywords

computer vision, deep learning, convolutional networks, object candidates, object detection, semantic segmentation, big data

Acknowledgements

First of all, I would like to appreciate the support that I have been given from my advisors. To Amaia, for her willingness to help, supervising the whole development of the project and her always reasonable advise. To Xavi, for keeping me motivated at every stage of the project, specially when things turned out bad, and for giving me the chance to take part in such an attractive project.

Obviously, the support always received from my family has been very important. Although they couldn't help me in a technical way, they have always shown interest in my work and have been there acknowledging my effort.

To the PhD candidates from the office, who have made my stay at the Image Processing Group (GPI) department unforgettable.

I could not end this section without expressing my thanks to Albert Gil and Josep Pujal, the software experts from the GPI, who always find a way to solve technical problems, no matter the difficulty and the time invested.

Contents

1	Introduction	1
1.1	Motivation	1
1.2	Thesis Outline	2
2	State of Art	3
2.1	Convolutional Neural Networks	3
2.2	Object Detection	5
2.2.1	Object Detection Networks	5
2.2.1.1	R-CNN	5
2.2.1.2	Spatial Pyramid Pooling	6
2.2.1.3	Fast R-CNN	6
2.3	Semantic Segmentation	7
2.3.1	Semantic Segmentation Networks	8
2.3.1.1	Simultaneous Detection and Segmentation (SDS)	8
2.3.1.2	Solutions based on Fully Convolutional Networks	11
3	Requirements	13
4	Design	14
4.1	Datasets	14

4.1.1	Pascal Visual Object Classes	14
4.1.2	Microsoft COCO	15
4.1.3	TRECVID Instance Search	15
4.2	Fine-tuning CNNs	16
4.2.1	Adaptation of CaffeNet to global Pascal VOC	16
4.2.2	Adaptation of SDS to local Microsoft COCO	17
4.3	Exploring local CNNs for Instance Search	18
5	Implementation	20
5.1	Software	20
5.2	Fine-tuning CNNs	21
5.2.1	Adaptation of CaffeNet to global Pascal VOC	21
5.2.1.1	Data preparation	21
5.2.1.2	Network definition	21
5.2.1.3	Fine-tuning parameters	22
5.2.1.4	Fine-tuning	22
5.2.2	Adaptation of SDS to local Microsoft COCO	23
5.2.2.1	Data preparation	23
5.2.2.2	Network definition	25
5.2.2.3	Fine-tuning	26
5.3	Exploring local CNNs for Instance Search	26
6	Experimental Results	28
6.1	Results on Pascal fine-tuning	28
6.2	Results on TRECVID	29
7	Conclusions	35

7.1	Future Work	35
-----	-----------------------	----

List of Figures

2.1	CNN main structure	4
2.2	Each row corresponds to the evolution of the layer's strongest activation given random subset of features. First and second layers' activations correspond to textures and edges, while the upper ones show more detailed, class-specific information. Also, note that upper layers only develop after a high number of epochs.	5
2.3	The spatial bins that form each grid have sizes proportional to the image size, providing fixed-length vectors which are fed to the fully connected layers	6
2.4	(Left) original image (Center) object segmentation, where instances of the same object are differentiated (Right) semantic segmentation output, which lacks information about the number of instances. As all of them are labeled with the same category, they are filled with the same color.	7
2.5	SDS Pipeline	8
2.6	Proposals generation scheme	9
2.7	Two-branch network architecture. (1) Input of the network, which is every object candidate generated at the first step . (2) Branch that extracts a set of features from the bounding box containing the object. (3) Branch that performs the feature extraction from only the region. (4) Concatenation of the two set of features	10
2.8	Left: original image. Center: region before refinement. Observe the undershooting (parts of the object missing) and the overshooting (random/weird stuff that is included despite not being part of the object). Right: refined region	11
4.1	Two examples of Microsoft COCO images and their ground truth at pixel level.	15

4.2	Example of the provided information for one of the TRECVID queries.	16
4.3	The basic pipeline for an image retrieval system.	19
5.1	Example of the text file that needs <i>Caffe</i>	21
5.2	(left) original architecture, (right) in red, the modifications introduced	22
5.3	Window file format	23
5.4	Part of the window file generated for the SDS fine-tuning on COCO. Observe (a) the image dimensions, 480×640 , (b) the number of object candidates (2008) and (c) the class from the first object with its overlap and bounding box coordinates. Note that at the very end of the line there is the number indicating the number of the object candidate out of the total	24
5.5	Note (a) the polygonal segmentation, which provides a not very accurate annotation and (b) how two annotated regions overlap .	25
5.6	(column 1) <i>CaffeNet</i> top ranking, (column 2) SDS ranking and (column 3) final ranking, which is composed by the shots in column 2 that appear in column 1.	27
6.1	Histogram of the training set and the two validation sets. In blue, the training set. In red, the small validation subset (v1717) and in green the big validation set (v5823). The amount of images per class is represented in % to provide a better visualization . .	29
6.2	F-score for each category. (blue) v1717, (red) v5823	30
6.3	Average Precision for each one of the queries using CaffeNet, Fast R-CNN and SDS.	31
6.4	Qualitative results for query 9073. The first row includes the example images with the localization of the query. Then, the second section contains the top ranking using SDS, and the third one shows the same using Fast R-CNN.	32
6.5	Qualitative results for query 9086. The first row includes the example images with the localization of the query. Then, the second section contains the top ranking using SDS, and the third one shows the same using Fast R-CNN.	33

6.6	Qualitative results for query 9088. The first row includes the example images with the localization of the query. Then, the second section contains the top ranking using SDS, and the third one shows the same using Fast R-CNN.	34
-----	---	----

Chapter 1

Introduction

1.1 Motivation

Computer vision is nowadays one of the most challenging lines of research in the wide field of artificial intelligence. One of the main reasons for this emergence is the *Big Data* problem, i.e. the huge amount of digital content that is generated every day. In the case of visual information, this trend has been highly motivated by the popularization of digital photography and online repositories. In the last decade, the arrival of smartphones have allowed people to take pictures and upload them to the Internet in just a matter of seconds. However, generating and storing these huge amount of data is useless if images are never seen or used again by users after they are uploaded. For this reason, there is a need to have tools which are able to automatically annotate them and retrieve them effectively from these large repositories.

Historically, these tasks have always been handled with *handcrafted* algorithms that provide image representations under the form of visual features. Histograms of colors or edges, or point-based descriptors such as SIFT or SURF have been central in computer vision during the first decade of 2000's.

However, in the last few years a massive switch has been experienced to describe visual content, from the handcrafted features to applying machine learning techniques to the actual design of the descriptors. This massive trend is popularly known as *Deep Learning* and is mainly based on Convolutional Neural Networks (CNN).

Deep learning features are at the core of the best solutions to most computer vision problems, such as face verification, scene understanding, motion estimation, saliency prediction or even sentiment analysis. It has also been introduced in the retrieval field, i.e. searching for images stored in large repositories. Actually, the popularization of smartphones has especially raised the interest in

retrieval for the case of a visual search, as many users will capture pictures of an object, location or person with their cameras and use them to formulate a query to a retrieval system.

This thesis addresses the visual search problem when the query is defined by an specific object in a scene. In particular, my research is aligned with the work that the Image Processing Group at the Universitat Politècnica de Catalunya (UPC) is jointly pursuing with another team at the Dublin City University (DCU). These two teams participated last Summer 2014 in the TRECVID Instance Search Challenge[19], a scientific challenge that faces a *big data* scenario: given a large collection of videos, retrieve the video shots that contain an instance of a given query (*e.g.* a non-smoking logo). The main goal of my project has been exploring new solution for this task based on state of the art of deep learning descriptors for objects. In this direction, my work compares the performance of SDS [8] with the one using *CaffeNet*, with the goal of assessing the importance of local information for such task. These results are compared with another local technique using Fast R-CNN [6] descriptors, which only make use of the bounding box information (no segments).

1.2 Thesis Outline

The thesis is structured as follows:

1. Chapter 2, "State of Art", reviews the related work released in the last few years focused on *deep learning* applied to object detection and semantic segmentation tasks.
2. Chapter 3, "Requirements", states the main contributions of this thesis.
3. Chapter 4, "Design", describes the stages followed during the development of the project to reach the final goal.
4. Chapter 5, "Implementation", specifies the technologies used and explains in more detail each one of the stages of the project.
5. Chapter 6, "Results", shows the results obtained for the different solutions proposed
6. Chapter 7, "Conclusions", overviews the whole thesis and evaluates the results obtained.

Chapter 2

State of Art

The related work in this thesis is centered in three fields. Firstly, an overview of the general studies on convolutional networks in computer vision is provided. Secondly, related works to the object detection task are presented, since they are the starting point of some methods present in the third part. Finally, the problem of semantic segmentation is reviewed because it is very similar in terms of features to the instance search case addressed in this thesis. The three presented parts offer the foundations upon which this thesis is supported.

2.1 Convolutional Neural Networks

Deep Learning visual systems are based on Convolutional Neural Networks (CNN), a structure that provides a framework to solve computer vision problems. By iteratively feeding images into a CNN, it is able to learn features of different hierarchical natures, as stated in [21].

Back in the 1990s, the concepts behind CNNs were already proposed by LeCun *et al.* [12], but they eventually fell out of fashion due to their high computation requirements and the popularization of Support Vector Machines (SVM). After two decades, Krizhevsky *et al.* [11] brought CNNs back to life thanks to his outstanding results at the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2012. Since then, the interest in Convolutional Neural Networks rekindled, and many researchers have succeeded at applying them in many other computer vision tasks.

One of the reasons that explain the recent dive into deep learning is the availability of large datasets with thousands or even millions of labeled examples (*e.g.* ImageNet), that are required to train these networks. In addition, powerful GPU implementations for deep neural networks have come out, such as NYU's *OverFeat* [17], Toronto's *cuda-convnet* [11] or Berkeley's *Caffe* [10].

CNN Structure. Though the architecture of the networks varies depending on the kind of problem it faces, most of them follow the same model. The typical structure consists of several convolutional layers followed by a lower number of fully connected layers. Figure 2.1 shows a basic CNN structure, where we see that after each convolution there is also a pooling layer, as well as a normalization and a rectification one, though they do not appear in this scheme. The final layer is by default a Softmax classifier, which provides an output vector of as many dimensions as the number of classes (*e.g.* a network trained for classifying ImageNet images will produce a 1000D output vector).

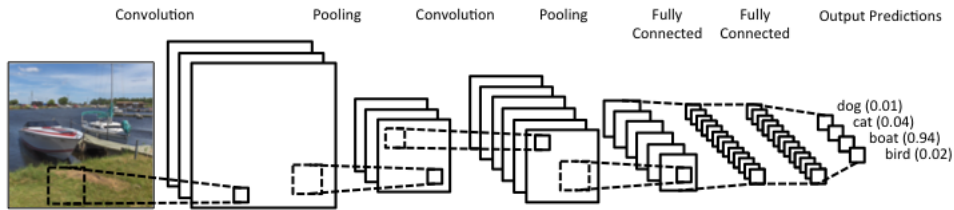


Figure 2.1: CNN main structure

Understanding CNNs. Despite their impressive results, it is still unclear why CNNs perform so well and how they can be optimally designed. With the aim of delving into this mystery, Chatfield *et al.* [2] presented a methodological study on a few configurations. In his work he tested *fine-tuning*, that is, an adaptation of a previously trained CNN to a novel domain. This is achieved by modifying the network last layer and extending its training from the already learned model weights. Also, Chatfield *et al.* replaced the classic Softmax classifier at the output of the CNN with a SVM trained on the layer before the last one. Finally, Chatfield also explored the effect of a dimensionality reduction on the dimensions of the features. Surprisingly, even a reduction from 4096D to 128D would imply a loss of only 2%.

Zeiler *et al.* [21] introduced a novel technique using a Deconvolutional Network [22] to visualize the feature activations on the ImageNet validation set. This way, the hierarchical nature of the features in each layer of a CNN was revealed leading to the conclusion that, while lower layers focus on edges, corners and textures, the upper ones capture more class-specific information, as figure 2.2 shows.

In the same line, Girshick *et al.* [7] performed an ablation study to understand which layers are critical for detection tasks, such as Pascal VOC [5]. The study revealed that features from fc_7 generalize worse than features from fc_6 , which means that the 29% of the CNN's parameters can be removed without degrading the results. Furthermore, if fc_6 is also removed, the results remain still quite good, concluding that much of the CNN's representational power comes from the convolutional layers, not from the fully connected ones.

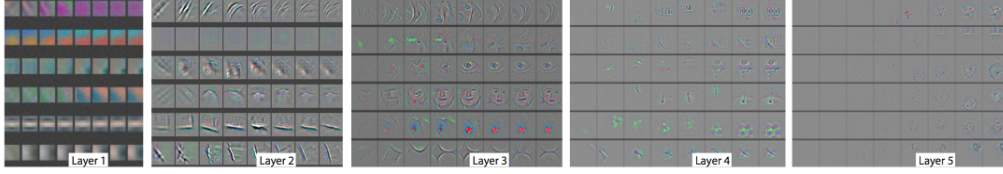


Figure 2.2: Each row corresponds to the evolution of the layer’s strongest activation given random subset of features. First and second layers’ activations correspond to textures and edges, while the upper ones show more detailed, class-specific information. Also, note that upper layers only develop after a high number of epochs.

2.2 Object Detection

Object detection is the process of finding instances of a specific category in an image. Object detection can be used in many real applications such as image retrieval or surveillance. Many works have been released setting the state-of-the-art in object detection by using deep learning descriptors generated with convolutional neural networks.

One of the most explored ways to face the challenge of object detection is using region proposals, also known as object candidates. These algorithms automatically generate a list of regions in the image which are candidate to represent an object, ranked from the highest to a lowest confidence and, in many cases, introducing diversity in the ranking. The regions provided by such algorithms can be used to generate a large amount of samples to train CNNs and, at test time, provide a reduced amount of locations in the image for analysis. MCG [1], Bing [4] and Selective Search [20] are some of the most commonly used techniques to generate object proposals.

2.2.1 Object Detection Networks

2.2.1.1 R-CNN

Girshick *et al.* [7] presented the technique combining object candidates and CNNs. In that case, object candidates were not pixel-wise segments but bounding boxes. Given an input image, a large number of region proposals are extracted using the *Selective Search* [20] method. After that, features from each region are extracted using a CNN to finally classify them using class-specific linear SVMs.

The CNN they used was a fine-tuned version of the popular AlexNet architecture [11] used for image classification. The network was trained on Pascal objects, which were generated from a training set of images using *Selective Search*. The object proposals with an intersection over union (IoU) higher than

a specified threshold were considered positives, remaining as negatives the rest of them.

Although R-CNN highly increased the state of the art for object detection, it is a very slow method, since each object proposal independently travels through the whole network to obtain its classification score.

2.2.1.2 Spatial Pyramid Pooling

With the aim of enhancing the R-CNN method in terms of speed and accuracy, He *et al.* proposed to compute the feature maps from the whole image only once to finally pool the features in arbitrary regions [9]. Thus, repeatedly computing the convolutional features for each region was avoided. This was achieved by replacing the last pooling layer of the convolutional stage with a spatial pyramid pooling layer, as illustrated in figure 2.3. Then, the convolutional layers are only trained using the whole image, and the selective search proposals are only used at the very end, and their features are pooled using the pyramid layer. This method yields a speedup of over one hundred times over R-CNN and is able to generate a fixed-length output regardless of the input image size.

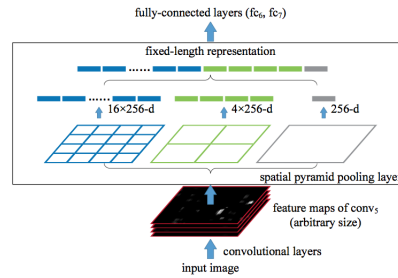


Figure 2.3: The spatial bins that form each grid have sizes proportional to the image size, providing fixed-length vectors which are fed to the fully connected layers

2.2.1.3 Fast R-CNN

With the R-CNN background and following the SPP method, Girshick proposed a faster version of its original work, which he called "Fast R-CNN" [6]. Unlike SPP, this new framework permitted all network layers to be updated during training, achieving remarkably better results on object detection with very deep networks like VGG16 [18]. Unlike SPP or R-CNN, Fast R-CNN did not use SVMs to train bounding box regressors or classification scores, but trained those functions at the same time directly during the network training, using a multi task loss.

2.3 Semantic Segmentation

The instance search problem addressed in this thesis can be understood as detecting and recognizing an object that is described by a region in a query image. Traditionally, solutions to instance search have discarded the precise pixel-wise information provided by regions and addressed the problem from a more global perspective, typically targeting the whole video frame as the basic work unit or, more recently, adopting a bounding-boxed representation of the object as a post-filtering stage. This classic strategy has been motivated by the large amount of storage and computation resources associated to a retrieval problem, as the one formulated in TRECVID Instance Search. However, in our work we aim at exploring a richer representation of the object, even if this is only used to re-rank a first list of shots generated by another faster approach. For this reason, this section of the State of the Art focuses on how objects are detected and recognized in the task of semantic segmentation.

The semantic segmentation task consists in assigning a category label to all pixels in an image (see Figure 2.4). This is already a challenging problem, yet it has a main limitation: semantic segmentation results show which objects appear in the image, but do *not* provide information about the number of instances of each object.

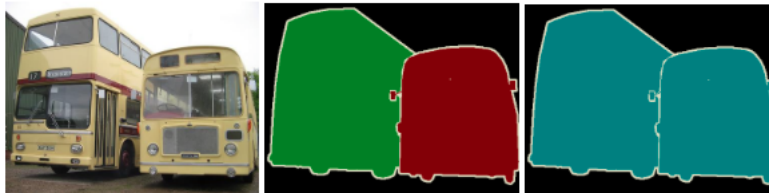


Figure 2.4: (Left) original image (Center) object segmentation, where instances of the same object are differentiated (Right) semantic segmentation output, which lacks information about the number of instances. As all of them are labeled with the same category, they are filled with the same color.

The scientific community addressing the problem of semantic segmentation developed a common evaluation platform that allows a fair comparison of the techniques. This benchmark was proposed in the framework of the Pascal Visual Object Challenge (VOC), containing a dataset of annotated images and a metric based on the Pascal Index, also known as Intersection over Union (IoU).

While all best performing techniques currently rely on convolutional neural networks, there exist two main approaches to the problem. The first one consists on obtaining a list of generic object candidates using an external algorithm, which are later semantically labeled with a CNN and combined to generate the segmentation in the image. On the other hand, other works consider the image as a whole and design a CNN that labels all pixels simultaneously, taking special consideration about the spatial correlations of these labels within the

image. This second family employs fully convolutional networks.

An important limitation related to the task of segmentation is the lack of images with pixel-level annotations. In contrast, it is easy to obtain large datasets with images labeled at both image and object-level (bounding boxes). Interesting research has been done in order to achieve good results while dealing with this situation. Papandreou *et al.* [15] studied several methods to train a CNN using combinations of different image annotations. They concluded that, a semi-supervised training mixing a low number of pixel-level labeled images and a much higher number of image-level annotations produces quite good results.

2.3.1 Semantic Segmentation Networks

2.3.1.1 Simultaneous Detection and Segmentation (SDS)

The Simultaneous Detection and Segmentation (SDS) system proposed by Hariharan *et al.* [8] from Berkeley University has been the main reference model in our work, so the method is described in more detail than others to facilitate the understanding of future sections.

The principles of R-CNN were basically extended to region-based object candidates by [7], Hariharan *et al.* [8], obtaining in September 2014 the best performance in the Pascal VOC Challenge for semantic segmentation.

For the feature extraction, they also introduced significant changes. A two-branch network architecture was proposed (figure 2.5). The first branch is fine-tuned on bounding boxes of MCG candidates, while the second one is fine-tuned only on the regions themselves, *i.e.* the background masked out.

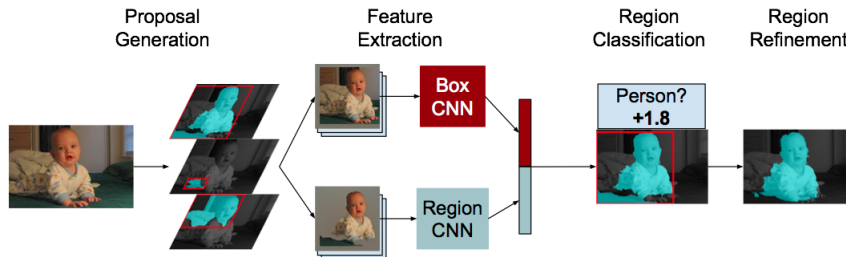


Figure 2.5: SDS Pipeline

Given an image, it is necessary to generate around 2000 region candidates (Figure 2.6). There are many methods that one can use to obtain region proposals, but they differ on the kind of output they produce, which can be either bounding boxes or/and segments. The SDS system is interested in obtaining segments, so Multiscale Combinatorial Grouping [1] is the one chosen by the authors to perform such task, as other methods like *Selective Search* [20] produce bounding boxes only.

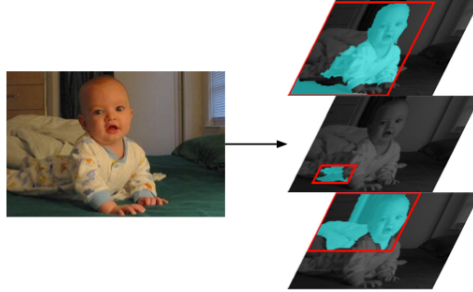


Figure 2.6: Proposals generation scheme

Feature extraction for boxes and regions

A convolutional neural network is used addressing this part of the pipeline and a novel architecture with two branches, also called pathways, is presented:

1. The first branch extracts features from the bounding box of a given object candidate.
2. The second branch extracts the features from only the region itself, *i. e.* subtracting the background.

Before passing through the CNN, each box and region is padded, cropped and warped to a square, as the network requires a fixed size. For both box and region networks, a set of features is extracted from the penultimate fully connected layer, leaving the last one (Softmax classifier) unused. The features from both branches of the network are finally concatenated and produce the resulting feature vector that will be used at the classification stage (Figure 2.7).

Given the architecture explained above, Hariharan *et al.* proposes three different strategies for fine tuning and extracting features:

Strategy A: Used as baseline: both branches (networks) are fine-tuned on bounding boxes. Thus, extracting features from the region foreground in the second pathway is suboptimal.

Strategy B: The second branch is fine-tuned on regions, but as the two networks are trained separately (like in A) it is still not the best solution.

Strategy C: Both networks are trained as a whole. It achieves the best results (Table 2.1)

Taking this into account, for the development of this project we will use only network C to extract features.

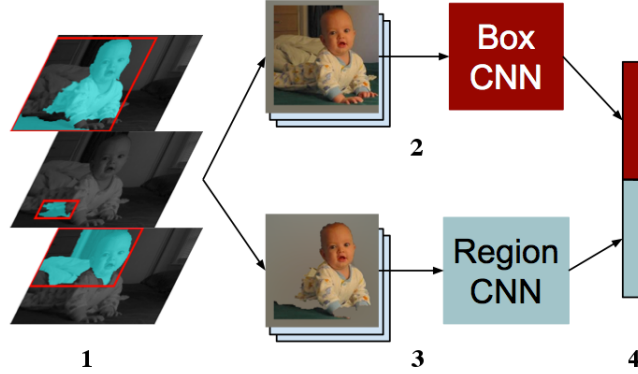


Figure 2.7: Two-branch network architecture. (1) Input of the network, which is every object candidate generated at the first step . (2) Branch that extracts a set of features from the bounding box containing the object. (3) Branch that performs the feature extraction from only the region. (4) Concatenation of the two set of features

	A	B	C
mean AP	42.9	47.0	47.7

Table 2.1: Results on average precision (AP), based on segmentation overlap, on VOC2012 val. Results are %

Classification of Object Candidates

Once we have the feature vectors for all the training boxes, a linear SVM is trained in order to classify each region. Firstly, an initial SVM is trained considering the following:

- Ground truth boxes are used as positives.
- Regions that overlap by less than a 20% with the ground truth are used as negatives.

Once the training is done, a new positive set is re-estimated:

- For each ground truth region, the highest scoring MCG candidate that overlaps by more than 50% is picked.
- Ground truth regions for which no such candidate exists, *i. e.* there is not a single candidate overlapping by more than 50%, are discarded.

After the new positive set is completed, the SVM is retrained. This process is found to provide better results than the first training.

Segmentation Refinement

Finally, a refinement for each region is performed (Figure 2.8). The reason yields in the fact that the region candidates obtained at the very beginning of the pipeline are created by a bottom-up process which makes no use of category-specific shape information. Thus, region proposals are prone to undershooting and overshooting.

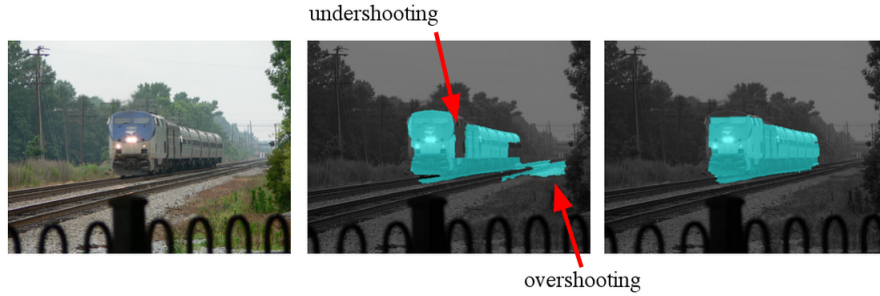


Figure 2.8: Left: original image. Center: region before refinement. Observe the undershooting (parts of the object missing) and the overshooting (random/weird stuff that is included despite not being part of the object). Right: refined region

As the relevance of this last step is only remarkable when facing segmentation tasks, we will skip it in our project.

2.3.1.2 Solutions based on Fully Convolutional Networks

Despite the good results obtained with SDS, which uses object candidates, other works have shown that directly feeding the whole image into the network is enough to produce state-of-the-art results and even improve them.

It must firstly be noticed an important difference between these techniques and the ones based on object candidates. While fully convolutional networks actually succeed in providing a pixel-wise labeling of the image, they fail into distinguishing among the different instances of an object category. So, for example, an image depicting a crowd of people may be correctly labeled at the pixel level as *person* by a fully convolutional network, but it will not be able to count the amount of persons.

Fully Convolutional Networks Long *et al.* [14] from Berkeley University demonstrated that CNNs by themselves are capable to exceed state-of-the-art results in semantic segmentation and introduced the term of "*Fully Convolutional*" networks. This architecture converts the fully connected layers into convolutional layers, while the last classifier layer is removed. Thus, the output is a bi-dimensional feature map, instead of a single vector. Then, a 1×1 convolution with channel dimension 21 was appended at the end to predict scores for each of the PASCAL classes (including the background) at each of the feature map location. Finally, the dense prediction (semantic segmentation) of

the input image is obtained by assigning to each pixel the label of the class with the highest score. This approach dramatically improved previous results in semantic segmentation, outperforming R-CNN and SDS.

DeepLab-CRF Following the concept of Fully Convolutional Networks [14], Chen *et al.* presented their "DeepLab" system [3] to significantly boost state-of-the-art on segmentation by localizing segment boundaries and produce more accurate outputs. As in [14], they convert a convolutional network into a fully convolutional one. Finally, the output map is up-sampled by bi-linear interpolation and a fully connected Conditional Random Field (CRF) is applied to refine the segmentation result. With his "DeepLab" system, Chen achieved the top result in the Pascal VOC 2012 at the moment this thesis was carried out.

Chapter 3

Requirements

During Summer of 2014, a team composed by researchers in Universitat Politècnica de Catalunya and Insight Centre of data Analytics of the Dublin City University took part in the TRECVid Instance Search Challenge. The main goal of this challenge is to build systems that are able to retrieve the video shots from a large video collection in which an instance of a specific category appears. There are 30 different categories, which can be either people, locations or objects. The DCU-UPC team addressed the problem by combining object candidates and off-the-shelf descriptors generated with *CaffeNet* (a slight modification of *AlexNet*), the convolutional neural network that Krizhevsky used in [11]. As the goal of the task was finding specific objects, it made sense to use algorithms to produce region proposals for each image.

The first problem of their system was that, although they wanted to describe objects, task that perhaps required a local solution, they used *CaffeNet*, which had been trained at global scale (to solve the ILSVRC). In this direction, my first contribution to improve their results is switching from the global *CaffeNet* to a convolutional neural network trained at local level. In this line, we have focused on the work released by Hariharan *et al.*: Simultaneous Detection and Segmentation [19]. In September of 2014, their system, which used a CNN trained at local scale, was at the top of the Pascal VOC detection task.

Taking into account the 1,000 ImageNet categories and its 1 million images training set, SDS was fine-tuned on a relatively small dataset, Pascal, which has ~6k training images and only 20 categories. Therefore, my second contribution aims at retraining the SDS network in a much larger annotated dataset: Microsoft COCO [13]. This new dataset has more than 80k training images and 80 classes.

Chapter 4

Design

This chapter describes the stages followed during the development of the project in order to achieve the final goal. The work has been divided in two main tasks: (a) fine-tuning the SDS network with the object masks contained in the Microsoft COCO dataset, and (b) assessing the available locally oriented CNN descriptors for the instance search task defined by TRECVID.

This chapter firstly presents the different datasets that were used during this project. Then, it introduces the design of the fine tuning experiments and the application of local descriptors to the TRECVID Instance Search task.

4.1 Datasets

4.1.1 Pascal Visual Object Classes

For the first stage of the fine-tuning part the Pascal dataset of 2012 was used. This dataset is commonly used for object detection and semantic segmentation tasks, and has the following features:

- It includes images of 20 different object categories.
- It is multi-label (*i.e.* an image may be labeled with more than one category).
- It is divided in 5717 training images, 5823 validation images and 5585 test images.
- It provides ground truth at pixel and bounding box level regarding object instances and object classes.

4.1.2 Microsoft COCO

The Microsoft COCO (Common Objects in Context) dataset was chosen to fine-tune the SDS network in order to generate better descriptors addressing the instance search task, as it is a much larger dataset with more images and categories in comparison to Pascal. Figure 4.1 shows two examples of images in the dataset and the provided ground truth for them, at pixel level.

This dataset has the following features:

- It contains 80 object categories.
- It is multi-label.
- It is divided in 82783 training images, 40504 validation images and 40775 test images.
- It provides ground truth for all images at pixel and bounding box level.

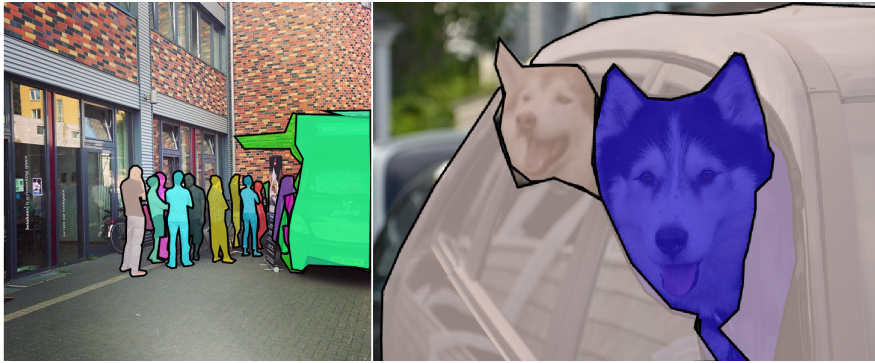


Figure 4.1: Two examples of Microsoft COCO images and their ground truth at pixel level.

4.1.3 TRECVID Instance Search

For the second part of the project, which consists in assessing the local CNN descriptors for the TRECVID Instance Search task, the TRECVID Instance Search dataset was used. This dataset is composed by a collection of videos from the sitcom Eastenders which are divided in shots, from which keyframes were extracted (with a frame rate of 1/4 fps).

The goal of this task is to rank the shots according to their similarity to different visual queries, which can be either be objects, locations or people. Every year they provide 30 different queries, which are represented by four images that contain it, along with pixel wise ground truth about where the instance is located in the image (see figure 4.2 for an example).

Due to the complexity and size of the target dataset (it contains more than 400.000 shots), for this work we have selected a subset of the provided data, which is composed by 13.386 shots (23.613 frames).

Category: Object

Description: a circular 'no smoking' logo

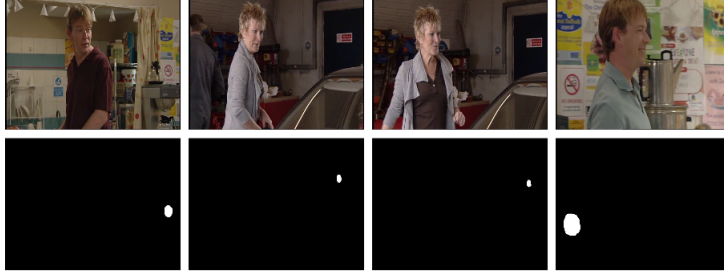


Figure 4.2: Example of the provided information for one of the TRECVID queries.

4.2 Fine-tuning CNNs

4.2.1 Adaptation of CaffeNet to global Pascal VOC

CaffeNet is a convolutional neural network trained to extract features at global scale, for instance, for image classification tasks such as the ImageNet Large Scale Visual Recognition Challenge. As previously stated, one of our contributions is switching to a CNN that extracts features at local scale, as the TRECVID task focuses on particular objects, not on the whole image. In order to achieve such target, this section presents a primary step to become familiar with the *Caffe* software [10], the framework employed to extract features from a convolutional neural network.

The task addressed as a first step was classifying Pascal images using *CaffeNet*. In order to do this, it is necessary to fine tune the network, by modifying the last layer (the one prior to the softmax layer) accordingly to the number of classes of our dataset. Thus, in this experiment we adapt *CaffeNet*, which is originally trained on the 1,000 ImageNet classes, to the 20 categories of Pascal. In order to complete this task, it was necessary to adapt the annotations provided by Pascal to a readable format for Caffe.

This first experiment posed the main challenge of the task, as the Pascal dataset is multi-label, which means that an image may contain instances of more than one category. In the official *Caffe* website¹ there is information about how to fine-tune a network with images that have only one label but

¹http://caffe.berkeleyvision.org/gathered/examples/finetune_flickr_style.html

not for multi-labeled ones, which arose some question about how to deal and address the adaptation. However, as the interest at this point of the project was just getting used to the process of fine-tuning networks, a simple approach was proposed to deal with the multi-label problem: from all the possible labels an image may have, only one of them was picked.

To fine-tune a network, these steps were followed:

1. **Preparing the data** to a format acceptable by Caffe. In this case the format was a text file with an image path and its corresponding label per line. At the end of this step we obtained two text files: one for all the training images with their labels, and another for the validation subset.
2. **Modifying the network** architecture to fit the new dataset, *i. e.* setting the last layer output vector dimensionality to 20 instead of 1,000.
3. **Specifying the dataset** for both the training and the validation steps in the definition file of the network, by determining the path to the text files generated at the first step, which indicate the paths to the images to use to train and validate the network.
4. **Defining training parameters** such as the number of iterations and the learning rate. Several factors were taken into account when setting the number of iterations during the training stage:

$$iterations_{train} = \frac{epochs \times training\ samples}{batch\ size}$$

where:

- *epochs* refers to the number of times that each image passes through the network. When training, it usually has values between 50 and 100.
 - *training samples* is the total number of training images.
 - *batch size* is the number of images that are fed at once to the network.
5. **Defining validation parameters.** In this case only the test iterations parameter is modified:

$$iterations_{val} = \frac{validation\ samples}{batch\ size}$$

6. **Launching fine-tuning** with a specific Caffe command, which takes as input: (a) the solver file including the parameters and definition of the network and (b) the initialization *CaffeNet* weights.

4.2.2 Adaptation of SDS to local Microsoft COCO

Fine tuning CaffeNet with Pascal images was an introductory step before addressing the main challenge in this thesis. So far, with the experiments carried

out, we learned how to fine-tune a convolutional neural network at global scale. But now the procedure changes quite a bit, due to the kind of data that the network receives: patches of the image instead of the whole image.

Therefore, at this point we have to use algorithms that generate object candidates for each image and label them as one of the categories of our dataset (or as background). Then, these labeled patches are fed to the SDS network as positive and negative examples for each class. This requires an adaptation of the format of the annotations provided by Microsoft COCO to a data structure acceptable by the software published by the SDS authors. There are two challenges to address: the actual format used for data storage to describe the bounding boxes and regions, but also switching from the 20 Pascal VOC classes to the 80 categories defined in Microsoft COCO.

The first task was addressed by intensively exploring the object candidates used on the original SDS code, which had been extracted from the Pascal dataset. The following chapter gives details on how this part was solved.

Once all the annotations from MS COCO were converted into the correct format for SDS, the next step was switching from the 20 Pascal VOC classes to the 80 categories defined in Microsoft COCO. Although we had experience fine-tuning (see subsection 4.1.1), doing the same with the SDS network was not straightforward, as in this case the fine-tuning needed to be performed at local-scale. In the "Implementation" chapter the variations that fine-tuning at local-scale introduces are explained. The main difference is found in the text file that is delivered to Caffe, which has a new format and needs to be generated in a different way.

When the text files were already prepared, the fine-tuning of the SDS network on Microsoft COCO was launched, but it failed due to some internal code problems which were out of our scope. Thus, the actual fine-tuning remains on stand by as future work.

4.3 Exploring local CNNs for Instance Search

As stated in the "Requirements" chapter, switching from a global network like *CaffeNet* to a CNN trained at local level was expected to improve the results obtained by the UPC-DCU team. In this line, locally-trained networks were used to extract deep local descriptors, which were later used to describe and rank the images in the TRECVID Instance Search task.

The pipeline followed to address this task is illustrated in figure 4.3.

Query set / Target Database. The query set consisted of 4 example images for each query, *i.e.* a total of 120 query images. The target database were 13386 shots which contained a total number of 23613 frames

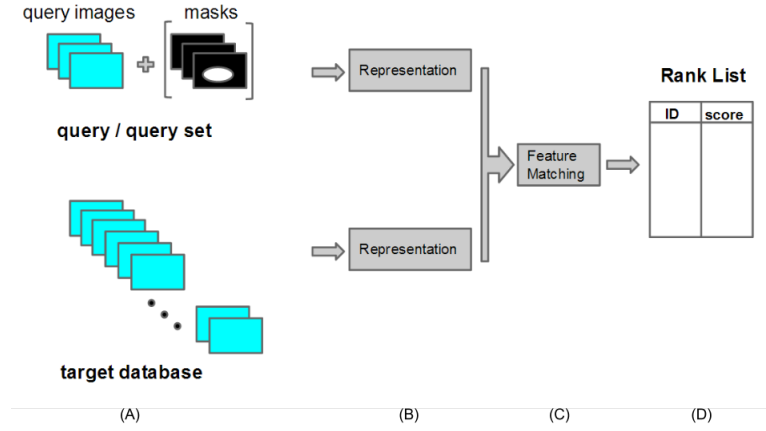


Figure 4.3: The basic pipeline for an image retrieval system.

Representation. We have 3 different CNN networks, which led to 3 different feature representations for our images:

1. Global CaffeNet descriptors, extracted from the penultimate fully connected layer
2. Local Fast R-CNN descriptors
3. Local SDS descriptors

Feature matching. The euclidean distance is the metric used to compare the query features with the database features

Ranking. Finally, a ranking listing the top 1000 shots (*i.e.* the 1000 with lower euclidean distance with respect to the query) for each query was generated.

Chapter 5

Implementation

5.1 Software

For all the procedures involving feature extraction using convolutional neural networks, the *Caffe* framework [10] was used. To exploit all its potential, GPUs are required. In our case the GPUs¹ from the UPC computing service accomplished that purpose.

The code to prepare the Pascal annotations to be suitable for the *CaffeNet* fine-tuning was written in Python and based on the documentation that Amaia Salvador prepared for her paper at the *CVPR ChaLearn Looking at People Workshop 2015* [16].

All the SDS code, as well as the scripts that needed to be written to adapt it to the COCO and the TRECVID datasets were developed in Matlab. The MCG candidates from Microsoft COCO required for fine-tuning at local scale were computed and stored by Jordi Pont, who was in fact one of the authors of the *Multiscale Combinatorial Grouping* [1] work.

Regarding the experiments with Fast R-CNN, the code and selective search object proposals were generated and provided by Amaia Salvador.

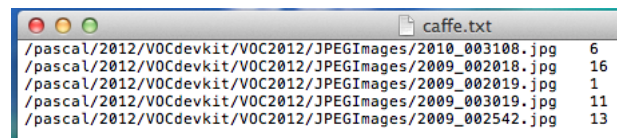
¹Nvidia's Tesla K20m, Titan Black, GTX980 and Titan Z

5.2 Fine-tuning CNNs

5.2.1 Adaptation of CaffeNet to global Pascal VOC

5.2.1.1 Data preparation

The first step was preparing all the data so that *Caffe* was able read it. The input to the network needs to be a text file with a path to the image an its corresponding label (see Figure 5.1).



```
/pascal/2012/VOCdevkit/VOC2012/JPEGImages/2010_003108.jpg 6
/pascal/2012/VOCdevkit/VOC2012/JPEGImages/2009_002018.jpg 16
/pascal/2012/VOCdevkit/VOC2012/JPEGImages/2009_002019.jpg 1
/pascal/2012/VOCdevkit/VOC2012/JPEGImages/2009_003019.jpg 11
/pascal/2012/VOCdevkit/VOC2012/JPEGImages/2009_002542.jpg 13
```

Figure 5.1: Example of the text file that needs *Caffe*

Pascal is a multi-label dataset, so the "problem" of assigning just one label per image had to be addressed. To do that, a Python dictionary was used. This data structure stores elements formed by a *key* and a *value*. The path of the image was considered to be the *key* while the label the *value*. The point was that a *key* can not be repeated in a dictionary, so each image would appear only once in the text file, with only one label, avoiding thus the multi-label trouble.

The script that performed such task received a text file with all the image names and their corresponding labels. Reading the file line by line, it created a new dictionary entry by assigning the image name to the *key* and the label to the *value*. If an image already existed in the dictionary, its *value* was overwritten. Therefore, at the end only one label per image was assigned, which was the pursued goal.

For this experiment, a partition of the train set of Pascal was performed to generate two new subsets of train and validation. Thus, the real validation set remained available to test the network once fine-tuned.

5.2.1.2 Network definition

The next step was modifying the *CaffeNet* architecture to adapt it to the 20 categories of Pascal. In this case only the last layer was changed, which is the most common practice when fine-tuning. Thus, the output of the last layer was switched from size 1,000 to 20. Note that the name of the layer had to be changed as well; otherwise, it would not be trained from scratch, which was the main purpose. In figure ... is shown only the definition file part that changed.

Along with these modifications, the text files generated in the previous subsection needed also to be specified and replace the ones written by default.

<pre>[...] layers { name: "fc8" type: INNER_PRODUCT bottom: "fc7" top: "fc8" inner_product_param { num_output: 1000 } } layers { name: "prob" type: SOFTMAX bottom: "fc8" top: "prob" }</pre>	<pre>[...] layers { name: "fc8_pascal" type: INNER_PRODUCT bottom: "fc7" top: "fc8_pascal" inner_product_param { num_output: 20 } } layers { name: "prob" type: SOFTMAX bottom: "fc8_pascal" top: "prob" }</pre>
---	--

Figure 5.2: (left) original architecture, (right) in red, the modifications introduced

5.2.1.3 Fine-tuning parameters

The last stage was specifying the parameters that the network should follow during the fine-tuning:

- *epochs* was set to 75, a mid value between 50 and 100, which are typical when training
- *training samples* = 4104
- *validation samples* = 1717
- *batch size_train* = 256
- *batch size_val* = 50

Taking this numbers and the definitions presented in the "Design" chapter into account, the resulting number of iterations for the training stage was approximately 1,200 and the network would be tested every 35 iterations to let the user know whether the fine-tuning is providing good results or not.

5.2.1.4 Fine-tuning

. At this point, an already implemented *train* method from *Caffe* was called. This command takes the specified training set in the definition file with its corresponding labels and starts retraining the network.

5.2.2 Adaptation of SDS to local Microsoft COCO

5.2.2.1 Data preparation

With the local fine-tuning being the new scenario, the steps to fine-tune the SDS network followed the same pattern that the global case, but with a remarkable difference when preparing the data. In this case, the text file did not have the *imagepath – label* format anymore, but the following one:

```
# image_index
img_path
channels
height
width
num_windows
class_index overlap x1 y1 x2 y2 n
```

Figure 5.3: Window file format

where:

- *image_index* is the ID of image.
- *img_path* are actually 3 lines, the first one containing an absolute path to the actual image, and the other two being paths to different pre-computed superpixel representations of the image.
- *channels* is the image color depth (typically 3).
- *height* and *width* are the image dimensions.
- *num_windows* is the number of object candidates computed for the current image.
- *class_index* is the class label.
- *overlap* is the overlap between the current object candidate (extracted using MCG) and its corresponding ground truth mask.
- *x1*, *y1*, *x2*, *y2* are the bounding box spatial coordinates.
- *n* is the number of object candidate out of the total.

An example of a text file generated with this format is shown in figure 5.4.

The first step to generate these text files was knowing the exact format of the object candidates used originally on SDS. By looking into them it was revealed that the MCG candidates for each image were defined by a structure with 4 parameters:

```
# 0
/imatge/efontdevila/seq/segmentation/COCO/tools/images/train2014/COCO_train2014_000000000009.jpg
/imatge/efontdevila/workspace/sds_eccv2014/COCO_cachedir/train/reg2spimgdir/COCO_train2014_000000000009.png
/imatge/efontdevila/workspace/sds_eccv2014/COCO_cachedir/train/sptextdir/COCO_train2014_000000000009.txt
3
480
640
2008
51 1.000 2 188 612 473 1
51 1.000 313 5 631 232 2
56 1.000 308 474 455 474 3
51 1.000 1 14 434 387 4
55 1.000 0 0 0 0 5
55 1.000 0 0 0 0 6
55 1.000 0 0 0 0 7
55 1.000 388 3 442 11 8
51 0.056 447 197 639 479 9
51 0.058 0 111 228 342 10
51 0.892 0 191 565 479 11
51 0.104 173 8 441 248 12
```

Figure 5.4: Part of the window file generated for the SDS fine-tuning on COCO. Observe (a) the image dimensions, 480×640 , (b) the number of object candidates (2008) and (c) the class from the first object with its overlap and bounding box coordinates. Note that at the very end of the line there is the number indicating the number of the object candidate out of the total

- **Bounding boxes:** a matrix of $N \times 4$, where N is the number of candidates previously extracted from the image, and 4 corresponds to the spatial coordinates that define the bounding box (up, left, down, right).
- **Superpixels:** a label matrix containing the superpixel partition of the image.
- **Labels:** a cell array that contains the superpixel labels that form each of the candidates.
- **Scores:** a $N \times 1$ vector that contains the scores of each of the ranked candidates ranging from 0 to 1 (where 1 means the highest chance of one region containing an object).

Since the precomputed candidates that we had from the whole COCO dataset had been obtained also using MCG, their format was the same as the one described above, yet the annotations from COCO were different from the Pascal ones.

The Pascal ground truth annotations are directly stored as a matrix with the same size as its corresponding image, where each pixel value is assigned depending on the class region it belongs to.

Instead, the Microsoft COCO annotations are stored using large JSON files that contain all the information for all the images of the dataset. In order to be able to read this information, a Matlab API is provided on the official COCO website². Perhaps the most noticeable feature of the COCO annotations is the way the segmentation of all the objects in each image is stored. Depending on the kind of annotation, it may be either stored as a polygonal segmentation or encoded via Run-Length Encoding (RLE).

²<http://mscoco.org/dataset/#download>

Taking into account this background, the main goal was converting the COCO annotations into the same format as Pascal, using the provided Matlab API. At this point, the inaccuracy of the COCO annotations became noticeable for two main reasons: (1) the polygonal segmentation provides not very precise boundaries and (2) some annotations overlap, *i. e.* a pixel may be labeled as two different objects (Figure 5.5).

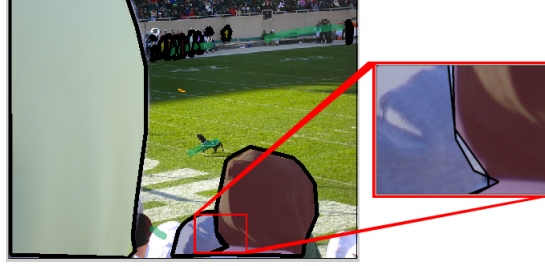


Figure 5.5: Note (a) the polygonal segmentation, which provides a not very accurate annotation and (b) how two annotated regions overlap

Specially the second statement posed some misunderstanding during the conversion of the COCO annotations. At this point the goal was getting the mask of each region and adding it to a zeroed matrix which had the same size as the corresponding image. The pixel values of the next region mask should be the same as the previous mask plus one, and so on. That algorithm worked well assuming that there was not overlapping between regions: each new region would fill only pixels with 0 value in the main matrix. However, if two regions overlap in one pixel, the value of that pixel would be the value of the region_1 plus the value of the region_2. That new value would not have any category assigned, leading to errors afterwards. That issue was solved by adding to the main matrix only those region pixels that would fall onto a zeroed cell.

Once this matter was clarified the window files were generated. To generate them, the previous step of obtaining the meta information of the object candidates' of each image had to be done, which took about a week of processing time to be completed. Note that we were extracting information from 2,000 region candidates per image and we had over 120k images (training set + validation set).

5.2.2.2 Network definition

The process followed here was the same as the one explained in the Pascal fine-tuning, but changing the *num_output* parameter of the definition file of the network to 81 (80+1) instead of 20, as 80 is the number of categories of Microsoft COCO and we had to take into account an extra "background" category for all those objects not belonging to any class. The name of the layer was set to "fc8_coco".

5.2.2.3 Fine-tuning

During the fine-tuning many problems appeared due to errors coming from the author’s code that were out of our scope. Therefore, the fine-tuning of the SDS network on COCO remains on stand by.

5.3 Exploring local CNNs for Instance Search

The steps below were followed to address the Instance Search task:

1. **Extracting the query descriptors.** We had 30 queries, each one with 4 example images, and the feature vector provided by the SDS network is 8192D. Thus, the descriptor obtained for each image had a size of 1×8192 , which was then saved to disk.
2. **Stacking the query descriptors** to produce a 120×8192 matrix containing all the features extracted from the queries.
3. **Extracting each frame’s features** and computing the euclidean distance between it and the query descriptors matrix. Note that for this part only 100 object candidates were picked from each image, to reduce the computational cost. Table 5.1 shows a study comparing the the computing time required to extract an SDS descriptor for an image depending on the number of object proposals.

number of candidates	computing time (s)
2000	32
1000	17
500	10
200	6
100	3
50	2,5

Table 5.1: Study comparing the invested computing time depending in the number of object candidates

As this part of extracting each frame’s descriptors and computing the distances would have taken about a day on a single GPU, we used 3 GPUs to reduce the time to 8 hours.

4. **Pooling distances for frames.** After getting the euclidean distance, the mean over all queries and the minimum distance over all the object candidates is acquired. At this point, the minimum distance between the current frame descriptor and each one of the 30 classes is obtained, as well as the object candidate that produced such distance. Only the resulting

distances array and the corresponding object candidates are stored to avoid disk space problems.

5. **Pooling distances of frames in the same shot.** Since the final goal of the TRECVID task is ranking the *shots* that are most likely to contain the query, it is necessary to average the distances obtained for all the frames of the shot, to get a single distance array that contains the minimum distance between the shot and each query
6. **Generating the final ranking** by sorting the shots according to their distance to each query. So the ranking will list, for each query, the top 1000 shots that are most likely to retrieve such query.
7. **Visualizing the results** to get a better understanding of what the system is doing. For each query, the top 12 shots are displayed, as well as the bounding box containing the object candidate that has matched the query descriptor the most.

To improve the results obtained following the pipeline described above, a re-ranking technique was performed. For each query a new ranking was generated listing all the shots, instead of only the top 1000. Then, this new ranking was compared to the ranking of the same query obtained last summer by the UPC-DCU team using the global *CaffeNet*. Finally, only those shots appearing in both rankings were kept. An example of that re-ranking is shown in the figure below:



Figure 5.6: (column 1) *CaffeNet* top ranking, (column 2) SDS ranking and (column 3) final ranking, which is composed by the shots in column 2 that appear in column 1.

Chapter 6

Experimental Results

6.1 Results on Pascal fine-tuning

We evaluate the fine tuned network on two image subsets: (1) the Pascal validation subset of 1717 images that was selected to use to evaluate the network during training and (2) the real validation set of 5823 images. We will refer to the first validation set as *v1717* and to the second one as *v5823*.

The results, evaluated using the classification accuracy (see Table 6.1), differ quite a lot depending on the image set that is used.

	v1717	v5823
accuracy (%)	59,31	4,14

Table 6.1: Results after fine-tuning on Pascal

The huge dissimilarity between the results required to be studied in detail. The first analysis focused on the number of images each class had, to reveal if the bad result was caused by a problem of unbalanced data. A histogram for each set of images (train, v1717 and v5823) was generated providing the results shown in figure 6.1.

It is really noticeable that the small validation set has a histogram almost exact as the one obtained for the training set. However, the histogram of the big validation set has a very different shape in comparison to the others. From this results we could extract a first conclusion: the difference in performance could be due to the unbalanced data, and the CNN was not trained enough for some specific categories.

The $f - score$ measure for each class was also computed and plotted on a graphic to get more information (figure 6.2). Looking at category 10, for instance, which was the one with more images, it can be observed that most of

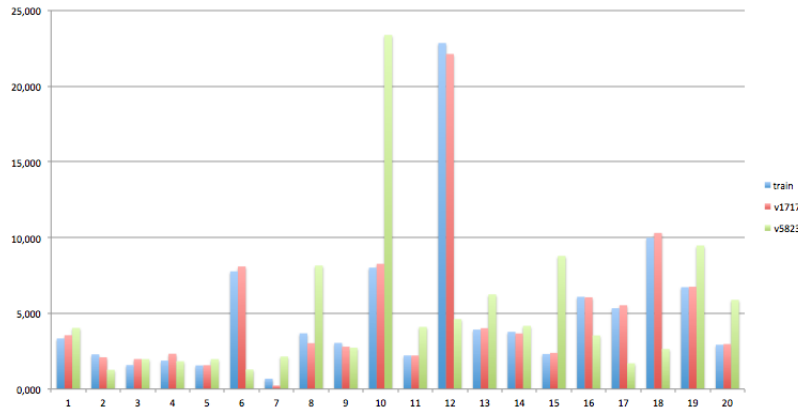


Figure 6.1: Histogram of the training set and the two validation sets. In blue, the training set. In red, the small validation subset (v1717) and in green the big validation set (v5823). The amount of images per class is represented in % to provide a better visualization

its images were not correctly classified as its f – score is very low. A similar behavior can be observed for most of the categories.

Along with this information, it was also stated that having images with multiple labels could have affected the results, since two images where the same objects appear could be labeled differently by the approach explained in section 4.2.1.

After this discussion, and having carefully analyzed the results, as the purpose of this step was only to get used to fine-tuning networks, it was decided not to continue exploring the reason of such bad results on the big validation set, and move forward to the next experiments.

6.2 Results on TRECVID

This section shows the results on the TRECVID subset using features from Object Detection CNNs. We compare the approach based on *CaffeNet* Fast R-CNN (trained on bounding boxes) and SDS (trained on bounding boxes and segments).

In figure 6.3 we compare the results obtained by using descriptors extracted from 3 different networks.

1. *CaffeNet*, which provides global descriptors
2. Fast R-CNN, which provides local descriptors
3. SDS, which provides local descriptors

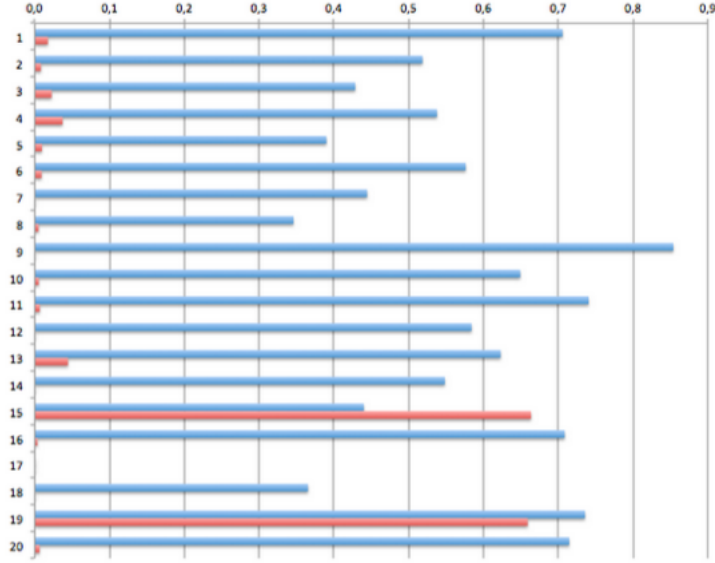


Figure 6.2: F-score for each category. (blue) v1717, (red) v5823

Results show that neither Fast R-CNN nor SDS improved last summer’s baseline. Only for some queries the results were better when using local networks. Specifically, SDS improved *CaffeNet* for 7 queries, while Fast R-CNN achieved it for 9. In terms of average precision over all queries, the SDS network slightly improved the Fast R-CNN results, and the re-ranking approach enhanced the results in both cases.

Figures 6.4 and 6.5 show the top 12 rankings of both SDS and Fast-RCNN for queries 9073 and 9086, for which SDS beat both *CaffeNet* and Fast R-CNN. It can be observed that the query object is correctly found in the frames using both techniques, but SDS localizes it better than Fast R-CNN, whose bounding boxes appear not to be very accurate. An explanation to that fact could be the algorithm used to extract the object candidates. Fast R-CNN uses *Selective Search*, which generates bounding boxes, while SDS is based on MCG, which generates segments. Thus, SDS may be able to mark the shape of the query better than Fast R-CNN.

On the other hand, it is also interesting to look at those queries for which the results were bad. For instance, the query 9088, which refers to a specific person. In figure 6.6 is shown that both SDS and Fast R-CNN retrieved shots where persons appear, but as they were not the query person, the shot was not valid. Clearly, both networks have been trained to detect persons, but they have not been trained to detect *that* person.

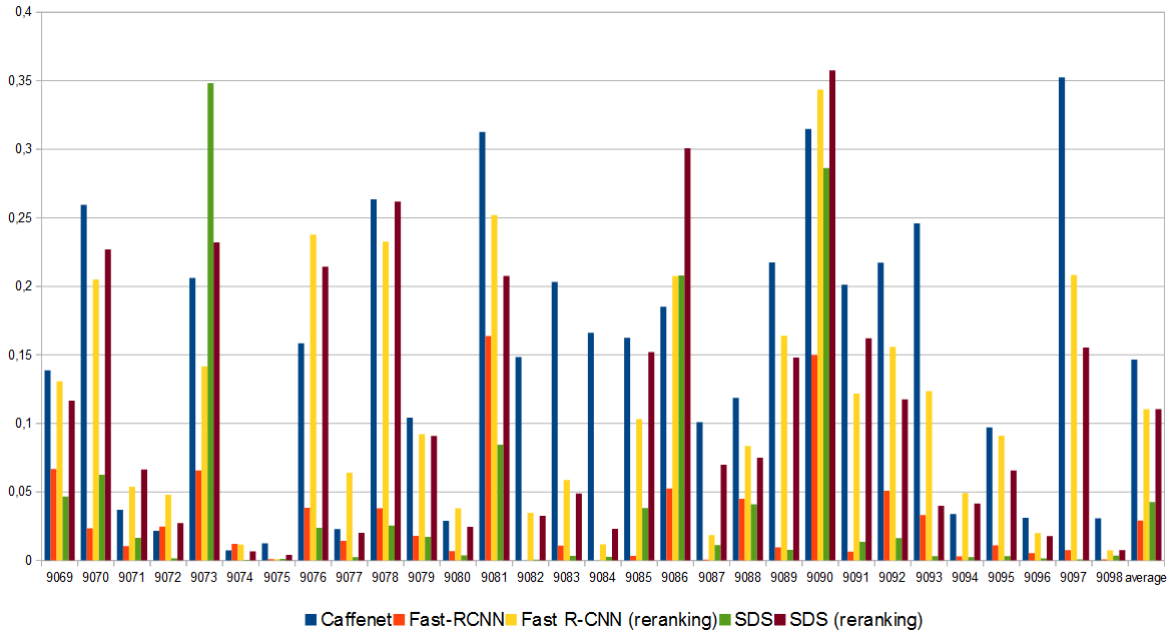


Figure 6.3: Average Precision for each one of the queries using CaffeNet, Fast R-CNN and SDS.



Figure 6.4: Qualitative results for query 9073. The first row includes the example images with the localization of the query. Then, the second section contains the top ranking using SDS, and the third one shows the same using Fast R-CNN.

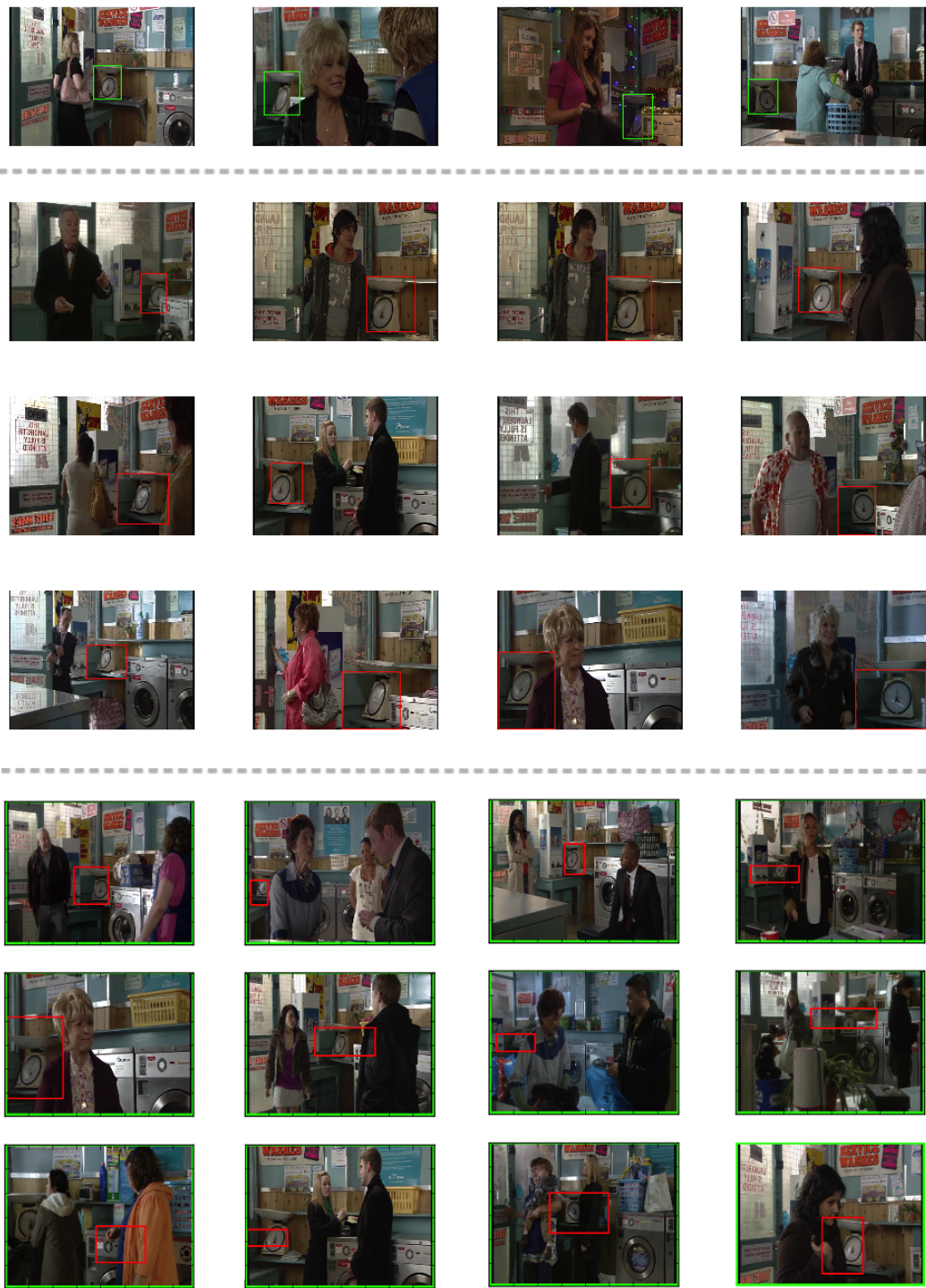


Figure 6.5: Qualitative results for query 9086. The first row includes the example images with the localization of the query. Then, the second section contains the top ranking using SDS, and the third one shows the same using Fast R-CNN.



Figure 6.6: Qualitative results for query 9088. The first row includes the example images with the localization of the query. Then, the second section contains the top ranking using SDS, and the third one shows the same using Fast R-CNN.

Chapter 7

Conclusions

This thesis has first presented guidelines on how to fine-tune convolutional neural networks using *Caffe* for the tasks of classification and object detection. Although the final results of fine-tuning SDS could not be obtained due to technical issues, all the necessary steps prior to running the fine-tuning were presented and rigorously analyzed.

The second contribution of this thesis has been exploring new solutions to improve the results obtained by the UPC-DCU team for last summer’s TRECVID Instance Search by using local deep learning descriptors.

The contribution of this work was reproducing their experiments but switching from the global *CaffeNet* to a convolutional neural network trained at local level. Despite that the local approach improved the results for some queries, the overall performance decreased. Many factors can explain this results. First of all, and perhaps the most important one: although the local networks that were used had been trained on objects, they had not been trained on the objects that they had to retrieve. This statement can be clearly supported with figure 6.6. The network detected a person, but not the *desired* person. Moreover, picking only 100 candidates per image decreased the likelihood of finding the query object. The fact that both SDS and Fast R-CNN networks did not improve the baseline *CaffeNet* results reinforces the idea that the descriptors they provided were not discriminative enough for the TRECVID classes, which are rather complex.

7.1 Future Work

For all these reasons, future work should aim at fine-tuning SDS with Microsoft COCO, and assessing its performance on the TRECVID dataset. It would be expected to find a significant gain when the network is trained for more (yet unrelated) classes, since the network would be more generic and robust.

Additionally, fine tuning SDS with TRECVID images should also be considered. Despite the lack of training data (only 4 images per query are provided in the dataset), using overlapping object candidates as well would increase the amount of data for fine tuning and could potentially boost the performance of our retrieval system.

When dealing with big datasets such as TRECVID, one needs to worry about the processing time as much as about performance. We have observed that SDS is very slow, and thus unfeasible to use with large datasets. However, many works have shown remarkable speedup for Object Detection Networks (such as Fast R-CNN or SPP), by learning convolutional layers on full images only and pooling features for the regions at the end of the network. In this direction, the same strategy could be applied to SDS, which would allow for a faster feature extraction. This way, the limitation on the amount of object proposals to use would not be necessary anymore, and one could expect to achieve much better results in less time.

Bibliography

- [1] Pablo Arbelaez, Jordi Pont-Tuset, Jon Barron, Ferran Marques, and Jitendra Malik. Multiscale combinatorial grouping. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 328–335. IEEE, 2014.
- [2] Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Return of the devil in the details: Delving deep into convolutional nets. *arXiv preprint arXiv:1405.3531*, 2014.
- [3] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. *arXiv preprint arXiv:1412.7062*, 2014.
- [4] Ming-Ming Cheng, Ziming Zhang, Wen-Yan Lin, and Philip Torr. Bing: Binarized normed gradients for objectness estimation at 300fps. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 3286–3293. IEEE, 2014.
- [5] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.
- [6] Ross Girshick. Fast r-cnn. *arXiv preprint arXiv:1504.08083*, 2015.
- [7] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 580–587. IEEE, 2014.
- [8] Bharath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. Simultaneous detection and segmentation. In *Computer Vision–ECCV 2014*, pages 297–312. Springer, 2014.
- [9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *Computer Vision–ECCV 2014*, pages 346–361. Springer, 2014.
- [10] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the ACM International Conference on Multimedia*, pages 675–678. ACM, 2014.

- [11] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.
- [12] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.
- [13] 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 *Computer Vision–ECCV 2014*, pages 740–755. Springer, 2014.
- [14] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *arXiv preprint arXiv:1411.4038*, 2014.
- [15] George Papandreou, Liang-Chieh Chen, Kevin Murphy, and Alan L Yuille. Weakly-and semi-supervised learning of a dcnn for semantic image segmentation. *arXiv preprint arXiv:1502.02734*, 2015.
- [16] Amaia Salvador, Matthias Zeppelzauer, Daniel Manchon-Vizuete, Andrea Calafell, and Xavier Giro-i Nieto. Cultural event recognition with visual convnets and temporal models. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2015 IEEE Conference on*. IEEE, 2015.
- [17] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013.
- [18] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
- [19] Alan F. Smeaton, Paul Over, and Wessel Kraaij. Evaluation campaigns and trecvid. In *MIR '06: Proceedings of the 8th ACM International Workshop on Multimedia Information Retrieval*, pages 321–330, New York, NY, USA, 2006. ACM Press.
- [20] Jasper RR Uijlings, Koen EA van de Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *International journal of computer vision*, 104(2):154–171, 2013.
- [21] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *Computer Vision–ECCV 2014*, pages 818–833. Springer, 2014.
- [22] Matthew D Zeiler, Graham W Taylor, and Rob Fergus. Adaptive deconvolutional networks for mid and high level feature learning. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2018–2025. IEEE, 2011.