Masknet: An Instance Segmentation Algorithm

Leveraging Object Detection and Semantic Segmentation to tackle Instance Segmentation

Master's thesis in Systems, Control and Mechatronics

JULIANO PINTO

Department of Signals and Systems
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2017

Master's thesis EX043/2017

Masknet:
An Instance Segmentation Algorithm

Leveraging Object Detection and Semantic Segmentation to tackle Instance Segmentation

JULIANO PINTO

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Signals and Systems
Chalmers University of Technology
Gothenburg, Sweden 2017

Masknet: An Instance Segmentation Algorithm
Leveraging Object Detection and Semantic Segmentation to tackle Instance Segmentation
JULIANO PINTO

Examiner: Fredrik Kahl, Signals and Systems Department
Supervisor: Måns Larsson, Signals and Systems Department

Master's Thesis EX043/2017
Department of Signals and Systems
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Inputs and corresponding instance segmentation outputs from the Masknet system.

Typeset in LaTeX
Printed by [Name of printing company]
Gothenburg, Sweden 2017

Masknet: An Instance Segmentation Algorithm
Leveraging Object Detection and Semantic Segmentation to tackle Instance Segmentation
JULIANO PINTO
Department of Signals and Systems
Chalmers University of Technology

Abstract

This thesis formulates, develops, and evaluates Masknet, a system that performs instance segmentation on images from the Pascal VOC 2012 dataset. Two main versions of the system are developed. The first one, simpler, uses an object detector CNN (convolutional neural network) to propose bounding boxes and classify objects in the scene. These bounding boxes are then fed into a mask proposal CNN, inspired by recent advances in mask generation, in order to generate binary masks for each instance.

The second version of the algorithm expands the mask proposal CNN to also use initial guesses for the binary masks, computed by a heuristic that uses the object detector CNN and also a semantic segmenter CNN. For this version, several different architectures are proposed and tested, and the best scoring one is finally deployed in the system. The best implementation of each of the versions is then preinitialized in a preliminary training step that uses data from the MS COCO dataset, trained on the Pascal VOC 2012 dataset and finally compared by computing the average precision achieved at each class of the dataset and the MAP^r (mean average precision throughout all classes) for the algorithm. The best scoring version achieves a MAP^r of 57.7% with an IoU threshold of 0.5.

Keywords: image analysis, instance segmentation, convolutional neural networks, object detection, semantic segmentation.

Acknowledgements

I want to thank my examiner, Fredrik Kahl, for his expert oversight of this thesis and the knowledge and encouragement he provided. I am also very grateful to my supervisor, Måns Larsson, for his unending patience, availability and motivation, and for the invaluable guidance given to me.

Additionally, many thanks to Olof Enqvist, for the great classes and for connecting me with Fredrik; and thanks to Pedro Diniz, for his diligent and skillful help with producing the images for this document.

Furthermore, I wish to express my sincere gratitude to all the people that provided me so much emotional support throughout this thesis. Thank you Lina, my life partner, for your unceasing encouragement, love and advice. You made my days better and my heart warmer. And thank you Mother, Father, Grandmothers Marley and Gilda, Brother and all of my family, for the lifelong support, inspiration and sincere belief that I could always achieve my greatest dreams, even if that includes moving to the other side of the world to follow my passion. I love you all.

Juliano Pinto, Gothenburg, June 2017

Contents

List of Figures

List of Tables

1

Introduction

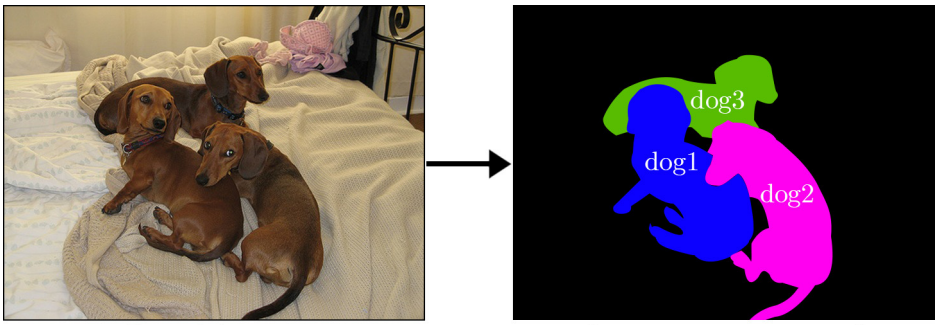
This thesis formulates, develops and evaluates two versions of an algorithm, baptized as Masknet, which performs instance segmentation on images from the Pascal VOC 2012 dataset. The first version uses a straightforward approach of predicting instances directly from object region proposals from an object detector CNN. The second version expands this idea by also using initial guesses for the masks of the instances to be generated, with the help of a semantic segmenter CNN. Both versions are developed and evaluated, and the results reported and discussed. All of the code developed for this thesis is made available at `https://github.com/JulianoLagana/masknet`.

1.1 What is instance segmentation, and why is it relevant?

A lot of progress has been made towards making machines that can better interpret and react to their environments. Televisions with gesture recognition [1], mobile phones that recognize the face of the user during use [2], personal robot assistants [3], and self-driving cars [4], are all examples of recent advancements the artificial intelligence field has witnessed. Following this trend, one of the major breakthroughs yet to come is making these machines able to visually interpret their surroundings with a quality comparable to human beings. Given the depth of this problem, the research community is first tackling smaller sub-problems that can yield insight into how to solve the original one. Sub-problems such as, for instance, image classification, and more recently, object detection and semantic segmentation, detailed in sections 2.2 and 2.3, respectively.

As progress in these problems evolved, a new task was popularized by [5]. This new task, later baptized "instance segmentation", is a much harder and more realistic problem than the former ones. The expected output for this task using a sample image from the dataset used in this thesis is shown in figure 1.1.

In this task, the algorithm is expected to output the precise locations, spatial extent and classes of all objects in the scene that are members of a set of predefined classes. Furthermore, and in contrast to semantic segmentation, the algorithm must be able to distinguish different instances of the same class in the scene. Hence, the name **instance** segmentation. For instance, if an image has three dogs in it, it's not



Input image Instance segmentation

Figure 1.1: Input and expected output from an instance segmenter output.

enough to simply denote which pixels belong to the "dog" class. It's necessary to also indicate which group of pixels belong to which dog in the image, making it inevitable to somehow deal with partial occlusions, and therefore to have a more complete understanding of the objects in the scene. Further details are provided in section 2.4.

Having an algorithm that can perfectly solve instance segmentation, or at least solve it as well as humans can, would greatly advance the way machines interact with their environment, allowing them to reason and adapt much better to different situations. A personal robot assistance could understand if it is in the kitchen or the office, if there is a person working in the room, if a person is asleep or if he/she fell on the ground. A self-driving car could better understand its environment. Knowing where possible pedestrians are and if the person in front is riding a bicycle or a motorcycle will help the car to make more informed decisions. Almost any machine that needs to interact with humans and has enough computing power could enhance its decision making by having such an algorithm to visually parse its surroundings.

This thesis is a contribution to the challenge of solving this image analysis task. It uses easily obtainable pretrained CNNs to solve the sub-problems of object detection and semantic segmentation, and with them, attempt to provide the instance segmentation of a scene.

1.2 Roadmap for the thesis

This thesis is separated into six chapters. The current and first one, where the problem being solved is stated, also contains an overview of related work, a brief overview of the main idea and finally an explanation of the datasets used and their contents.

After that, a chapter devoted to filling any knowledge gaps that might impair the reader's understanding of the ideas in this thesis is presented. In this chapter, the foundations of what is a convolutional neural network, and explanations of what are the problems of object detection, semantic segmentation and instance segmentation are given.

That chapter is followed by the Methodology chapter, which starts by providing a conceptual explanation of the main ideas of this thesis. This is then followed by a concrete explanation of their implementations, and a presentation of the training protocol used for optimizing the networks. Further, this chapter also compares the different architectures proposed for one of the versions of Masknet, and ends with a section explaining how a new step in the training protocol, which greatly improved results, was performed.

Subsequently, there is the Results chapter. This chapter is mainly focused on providing the results found for the algorithms developed. Both the results for the optimization of the mask proposal subsystem and the instance segmentation evaluation of the system as a whole are presented.

Finally, a Discussion chapter is introduced, where a visual assessment of the mask proposal subsystem's performance is presented, together with an analysis of the impact that each subsystem has to the final instance segmentation quality of the algorithm. This chapter ends with a section devoted to suggesting directions for further work that could improve the ideas presented here.

The thesis ends with a brief Conclusion chapter, which summarizes the main ideas and results obtained throughout the development of this project.

1.3 Related work

Ever since the "Simultaneous Detection and Segmentation" paper [5] that popularized the problem of instance segmentation was published, this task has been undertaken by a variety of different approaches. Most of the successful approaches use custom convolutional neural networks in some way, together with new systems and ideas, like recurrent neural networks [6], or conditional random fields [7].

An early, notable contribution proposed for this task was the Hypercolumns [8] paper. The authors acknowledged that using output only from the last layers in a CNN typically fails to provide fine spatial information that could allow precise localization. In order to amend that, they create a new type of pixel descriptor, named the hypercolumn at that pixel, which comprises of the vector of activations of all the CNN units above that pixel. By leveraging this pixel descriptor, they managed to improve the state-of-the-art MAP^r by nearly 21%.

Since then, the MS COCO detection competition (despite the name, the task is to perform instance segmentation) has been a powerful driving force for the problem at hand, and each year the competitors invent new ideas and push the state-of-the-art further. The winner of the 2015 iteration of this competition, "Instance-aware Semantic Segmentation via Multi-task Network Cascades" [9], in short MNC, proposed the idea of separating the task of instance segmentation into sub-problems, and designing one CNN for each step. The system is then jointly trained using a loss function that takes into account all steps of the process. The current thesis draws some inspiration from this success, namely that of trying to solve a problem by designing a specific network for each sub-problem.

In 2016, the proposal that won the MS COCO competition was the "Fully-convolutional Instance-aware Semantic Segmentation" [10], where the authors presented the first fully convolutional solution to the task (i.e. a single CNN capable of performing this task, from image input to image output). The authors argue that CNNs are translationally invariant by design, and hence develop a new, translationally variant layer to allow them to leverage spatial information. This new layer is an assembly layer, which joins feature maps that learned to predict position-sensitive inside/outside score maps into a single feature map, later used for the mask proposal for each instance.

Since then several new ideas have been presented [11, 12, 13, 14, 15], and one of the most recent and relevant ones is Mask R-CNN [16]. Coincidentally, this network also uses the idea proposed in this thesis of using an object detector to propose candidate regions with a high change of having an instance inside them, and then feeding these into a subsystem for mask prediction. However, the authors developed a way to extend a state-of-the-art object detector, Faster R-CNN [17], to perform both tasks in the same network, with a class-sensitive mask prediction step. Nevertheless, the idea of using initial guesses for the masks, based on object detection and semantic segmentation outputs, is still unique to this thesis. The best MAP^r score achieved (metric between 0 and 1 that evaluates the instance segmentation quality, 1 denoting perfection) across all variations of the proposed algorithm was 0.577 (a comparison between other state of the art algorithms is shown in table 4.4.

1.4 Brief explanation of main idea

This thesis develops and evaluates two main versions of an instance segmentation algorithm. In the first one, the input image is processed by a semantic segmenter CNN, and by an object detector CNN, to provide the semantic maps and bounding boxes for the later stages of the network.

These outputs are fed into a heuristic that aims to provide initial guesses for the masks of each region enclosed by the bounding boxes, such that pixels in the mask have a strong likelihood of belonging to the instance that needs to be segmented, but at the cost of consistently leaving parts of the desired instance unmasked. These masks are simply binary images in which pixels with the value 1 denote positions that belong to the instance, whereas pixels with the value 0 do not.

Together with the original patches of the image enclosed by each bounding box, the initial guesses for the masks, henceforth referred to as "partial masks", are then fed into another CNN, the mask proposal subsystem, which finally outputs the instances found in the scene. The idea is that these partial masks can possibly help the system to decide which instance to segment, if more than one is present in the region.

The second version, somewhat simpler, does not compute partial masks for each bounding box region. Instead, it uses only the input image and the bounding boxes obtained from the object detector CNN. The region enclosed by each bounding box is directly fed to the mask proposal subsystem, which then outputs the instances

found in the scene. In this thesis, both of these versions are developed, and later evaluated using the standard metrics for the instance segmentation problem.

1.5 Datasets used

Two datasets were used in this project, the Pascal VOC 2012 dataset and the MS COCO 2016 dataset. This section provides a brief overview of these two datasets, explaining their contents and provided ground truth data.

1.5.1 Pascal VOC

The Pascal Visual Object Classes challenge [18], first introduced in 2005, features in its latest release (the 2012 version) 2913 colored images of variable size, comprised of 6929 objects. A sample subset of the images present in the dataset is shown in figure 1.2. Ground truth data, generated by humans, is available for twenty different classes of objects, ranging from everyday objects you would find inside your house, like bottles, dining tables, television monitors, sofas; to objects usually found in very different contexts, such as sheep, motorbikes, boats and airplanes.

The ground truth data available for each image is comprised of:

1. Semantic segmentation of the scene.
2. Object segmentation of the scene (an image in which the value of each pixel denotes the object to which it belongs to).
3. An annotation for each object present in the scene, specifying its class, bounding-box, and other types of information not relevant to this thesis.

With such ground truth data, it's possible to know exactly which objects are in a given scene, where they are located, and their exact spatial extent. All of this information is used throughout this thesis to train and evaluate the developed systems.

Furthermore, the data is divided into training, validation and test sets. However, no ground truth data is available for the images in the test set, so that the competition is guaranteed to be fair (i.e. so as to make it impossible to train on images from the test set). The competitors submit the results to an evaluation server that ranks them automatically according to the metrics of the challenge.

Because of this, if one wants to evaluate an algorithm using a metric which the Pascal VOC evaluation server does not compute, then one is restricted to use only data from the training and validation set. This is indeed the case for this thesis, since the metric that evaluates the quality of instance segmentation, MAP^r, is not computed by the Pascal VOC evaluation server.

Figure 1.2: Sample images taken from the the Pascal VOC 2012 dataset.

1.5.2 MS COCO 2016

The Microsoft Common Objects in Context dataset [19], in short MS COCO, is a much larger and more varied dataset than the Pascal VOC 2012. Its latest version, 2016, contains more than 200 thousand images, in a variety of different settings, locations and spatial sizes. Ground truth data generated by humans is available for all of the images in the training and validation dataset, for the 80 different classes specified in the competition. A sample subset of the images from the validation set is shown in figure 1.3

The ground truth data available for each image is provided in a per-object basis, and each object has an annotated data structure which provides the following information:

1. Which image that object belongs to in the dataset.
2. The object's class.
3. A segmentation mask for that particular object, encoded into a custom RLE (run length encoding) scheme provided by the competition.
4. The bounding box for the object.
5. Several other types of information irrelevant to the present study.

Similarly to the Pascal VOC dataset, an evaluation server is also available for com-

petitors to upload their results and be automatically ranked using ground truth for the test data (not made publicly available). This evaluation server does provide a calculation of the MAP^r metric, but this was not used in this thesis since no instance segmentation evaluation was done using this dataset.

Figure 1.3: Sample images taken from the the MS COCO 2016 dataset.

2

Theory review

This chapter is dedicated to filling any knowledge gaps that might impair the reader's understanding of the ideas presented in the thesis. It first presents a brief overview of what Convolutional Neural Networks are, and how they differ from regular neural networks. After that, there is an explanation of what are the problems of object detection and semantic segmentation, and finally, an introduction to the problem of instance segmentation, subject of this thesis.

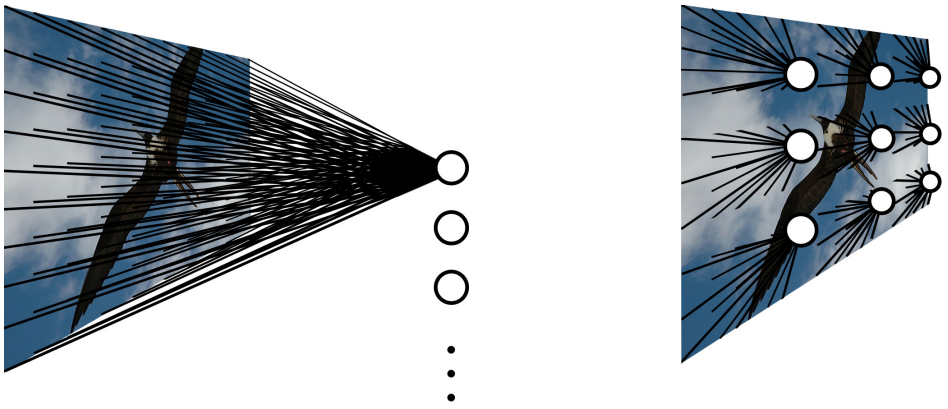
2.1 Convolutional Neural Networks

Artificial neural networks have been around for decades now, and excel at a multitude of previously unreachable tasks in machine learning. However, if one tries to use them in tasks related to images, one quickly realizes that they do not scale well for these types of problems. When the inputs are images, even more so if colored images, the amount of connections and hence learnable parameters increases drastically. For instance, if one is trying to learn a task to which the input is a 800×600 colored image, and the first layer has, say 1000 neurons, the amount of learnable parameters, only for this first layer, is more than 1.4 billion.

Because of this, if one aims to work with an image as input, it becomes necessary to either reduce the dimensionality of the data or reduce the amount of connections at each layer. Convolutional neural networks, in short, CNNs, are neural networks that restrict its neurons to connect to only a subset of the image's pixels at each time. This is accomplished by, instead of using fully connected layers, using convolutional layers, in which the image is convolved with a series of learnable filters.

For convolutional layers, all neurons are forced to use the same weights, and connect only to a small patch of the image (e.g. a 5x5 pixel area). Furthermore, they are organized in a spatial grid throughout the image, so that each one connects to a slightly shifted location from where the last one was, as illustrated in figure 2.1. Because of this arrangement, the output of this grid of neurons will be the convolution of the chosen weights with the input image. Hence, these chosen weights can be seen as a filter which is convolved with the image, generating the output.

At each convolutional layer there are usually several learnable filters of the same spatial size, and convolving the image with them produces several output images, where each one is denoted a "feature map". These several feature maps are then



(a) Fully connected layers, used in regular neural networks.

(b) Convolutional layers, used in convolutional neural networks.

Figure 2.1: The difference between regular and convolutional layers.

stacked in a 3D data object with dimensions $W \times H \times N$, where W and H denote the spatial dimensions of the data, and N is the number of filters used in the layer. This 3D data object can then be convolved again with a new set of filters, generating new outputs.

To quickly specify a convolutional layer size, the notation $W \times H \times D \times N$ is used, where W and H specify the spatial size of the filter to be used (i.e. the spatial extent in pixels of the area to which each neuron used in the convolution will connect to), D specifies in how many feature maps the neurons will connect to, and N denotes the number of learnable filters that that particular convolutional layer possesses.

Besides convolutional layers, convolutional neural networks also commonly use two other types of layers: ReLU layers and max-pooling layers. ReLU layers (Rectified linear unit layers), operate on a pixel-wise manner in their inputs, calculating the output for each pixel as

$$\mathrm{ReLU}(x) = \begin{cases} x, & x \geq 0 \\ 0, & x < 0 \end{cases}, \tag{2.1}$$

where x is the value of the input pixel. Max-pooling layers operate independently in each feature map of the input, reducing its spatial size by only keeping the maximum value found in a small region of the image, and discarding all others.

In practice, it's common to alternate convolutional layers with ReLU and max-pooling layers, to guarantee that the convolutions being done are non-linear, and to progressively reduce the size of the data, hence reducing even further the computational requirements for the network.

As the image gets convolved with these learnable filters, and the output is convolved with even more learnable filters, the network has the potential to discover which filters are able to extract the features from the image which are meaningful for the task at hand.

Therefore, one way to interpret this cascade of convolutional layers is as a feature

extractor, which after some layers produces a much smaller data object than the input image, but hopefully with still the same meaningful characteristics for the task at hand. Following this cascade of convolutional layers, it's common to add one (or more) fully connected layers, and a softmax layer, which will then try to regress the nonlinear function that maps the extracted features to the desired output.

These networks are then trained with the same backpropagation algorithm used for regular neural networks. The main difference is that now most of the weights are shared between neurons, so backpropagation is much more efficient. Furthermore, it's usually not possible to fit all training examples in memory, so stochastic gradient descent is regularly chosen for optimizing the weights. The interested reader is invited to read [20] for a thorough explanation of the adaptations needed.

2.1.1 VGG-16

One of the most iconic examples of a convolutional neural network is the VGG-16 network. First introduced in [21], this network was one of the deepest (highest number of layers) at the time. It achieved an error rate of 7.3% in the ILSVRC (ImageNet Large-scale Visual Recognition Challenge) 2014 competition, where the contestants are asked to predict the class of an image among a set of 200 pre-defined classes. It's success showed the research community that by increasing the depth of the networks, they could learn how to hierarchically represent visual data, and hence achieve previously unattainable results.

It uses 13 convolutional layers, alternating convolutions with max-poolings and ReLU layers. The convolutional layers are then followed by three fully connected layers, with 4096, 4096 and 1000 neurons, and lastly a softmax layer to predict the classes.

Given the successful results obtained by this network and the computational effort put into training it with the ImageNet dataset (this network was trained on 4 Nvidia Titan Black GPUs for up to three weeks), this CNN is usually used by other networks as a starting point for a feature extractor. The same strategy is performed in this thesis. Instead of trying to engineer a good network architecture, and then spending time and computing power to train it thoroughly, this thesis uses the VGG-16 network as the feature extractor for the images, and only fine-tunes its weights to the task at hand.

2.2 Object detection

Among the key open problems in image analysis, is the problem of object detection. This problem is the task of, based on an input image, predict the locations and approximate spatial extent of objects of a set of predefined classes, while also assigning the correct class to each of them. The location and spatial extent of each object is to be encoded as the coordinates of the tightest rectangle that encloses the entire object, usually denoted as a "bounding box". A sample output of an algorithm

solving this problem is shown in figure 2.2.

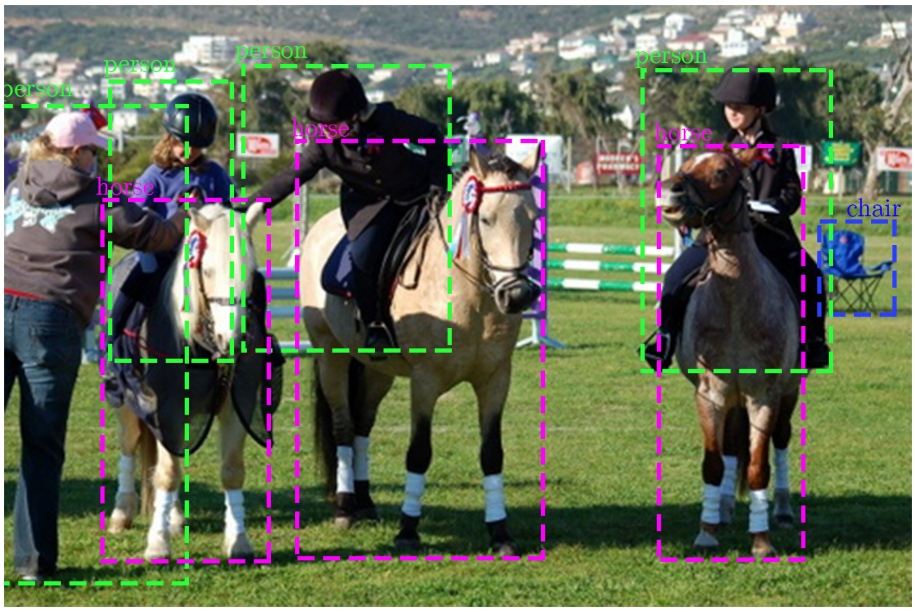


Figure 2.2: Example output from an object detector algorithm.

Although many different approaches were undertaken to try to solve this problem, the most successful one at the moment uses a special type of convolutional neural networks: region based CNNs. These networks work by first utilizing a separate method, usually class agnostic and only leveraging low level pixel information, to quickly propose a large amount (thousands) of region proposals (i.e. candidates for the predicted bounding boxes). These proposals are then warped into the expected input size of a feature extractor CNN — VGG-16 for instance — which outputs a feature vector for each region.

This vector is then fed to a set of linear support vector machines [22] (or more recently, a series of fully connected layers), in order to compute its classification among the predefined classes. This last step also computes a confidence score, which aims to rank the proposals so that only a subset of them can be returned. Further, the feature vector is also fed to a regressor that aims to improve the original proposal's bounding box quality.

Because of the jump in performance from other approaches, these region based CNNs are now the standard approach for object detection. Recent approaches have made them significantly faster [23, 17], making them even more useful to the research community. This project uses the region based CNN called Fast R-CNN. First introduced in 2015, this network is still capable of achieving competitive results on datasets like the one used for this project. The main difference of Fast R-CNN from other region based CNNs is that this version shares the computations of the convolutional layers between the different proposals, by computing one single feature map for the entire image and inferring the individual feature maps for each region from it. Further details can be found in the original paper [23].

2.3 Semantic segmentation

Another very important problem in image analysis is the problem of semantic segmentation. This problem refers to the task of assigning one of a pre-defined set of classes to each pixel location in an input image. In contrast with object detection, the notion of a single object is not needed anymore, but now the spatial extent of each class of objects must be predicted on a pixel level. A sample output from an algorithm performing this task is shown in figure 2.3.

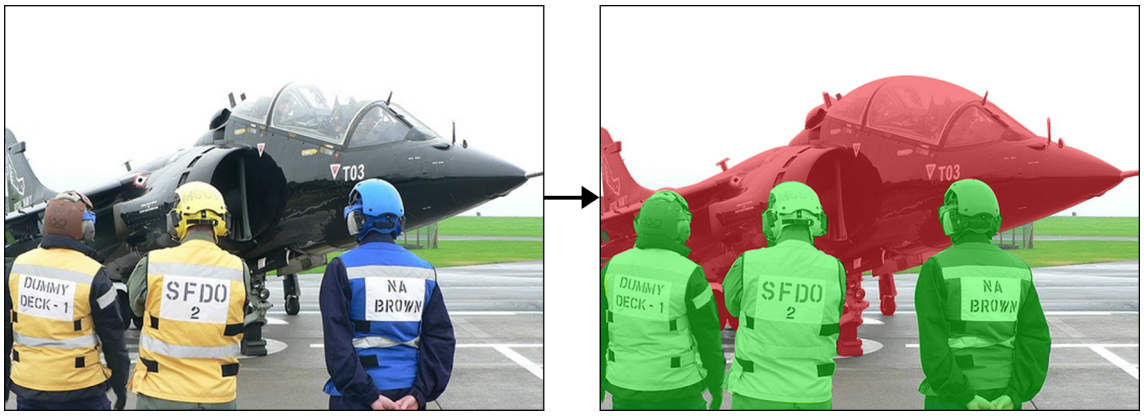


Figure 2.3: Example input and output from an algorithm performing the task of semantic segmentation. Pixels overlaid with the color green belong to the "person" class, and with the color red belong to the "airplane" class.

The major breakthrough for advances in this area was the paper entitled Fully Convolutional Networks for Semantic Segmentation [24]. This paper showed how to reinterpret the layers in a standard CNN and perform minor modifications to it to allow for variable size inputs, and images as outputs, instead of simple classifications as was done until then. Naturally, this new type of CNNs, namely, FCNs (fully convolutional neural networks) proved to be the best approach to spatially dense prediction tasks, such as semantic segmentation.

The aforementioned paper proposes three different FCN architectures to tackle the problem of semantic segmentation. The best performing one, FCN-8s, is the one used for this project. The main difference between the proposed architectures is how they combine semantic information from the last layers of the network with appearance information from the first layers. FCN-8s uses a skip architecture that connects the feature maps before the third and fourth pooling layers to the final layer, by upsampling them to the final layer's spatial size and summing both predictions. Further details can be found in the original paper [24].

2.4 Instance segmentation

The problem of instance segmentation can be understood as simultaneously solving object detection and semantic segmentation. Here, it's necessary to locate every

single object from a pre-defined set of classes, to predict its correct class, and also to provide a binary mask for each of them that specifies which pixels belong to that particular object and which do not. The difference between the output from a semantic segmenter and an instance segmenter is illustrated in figure 2.4.

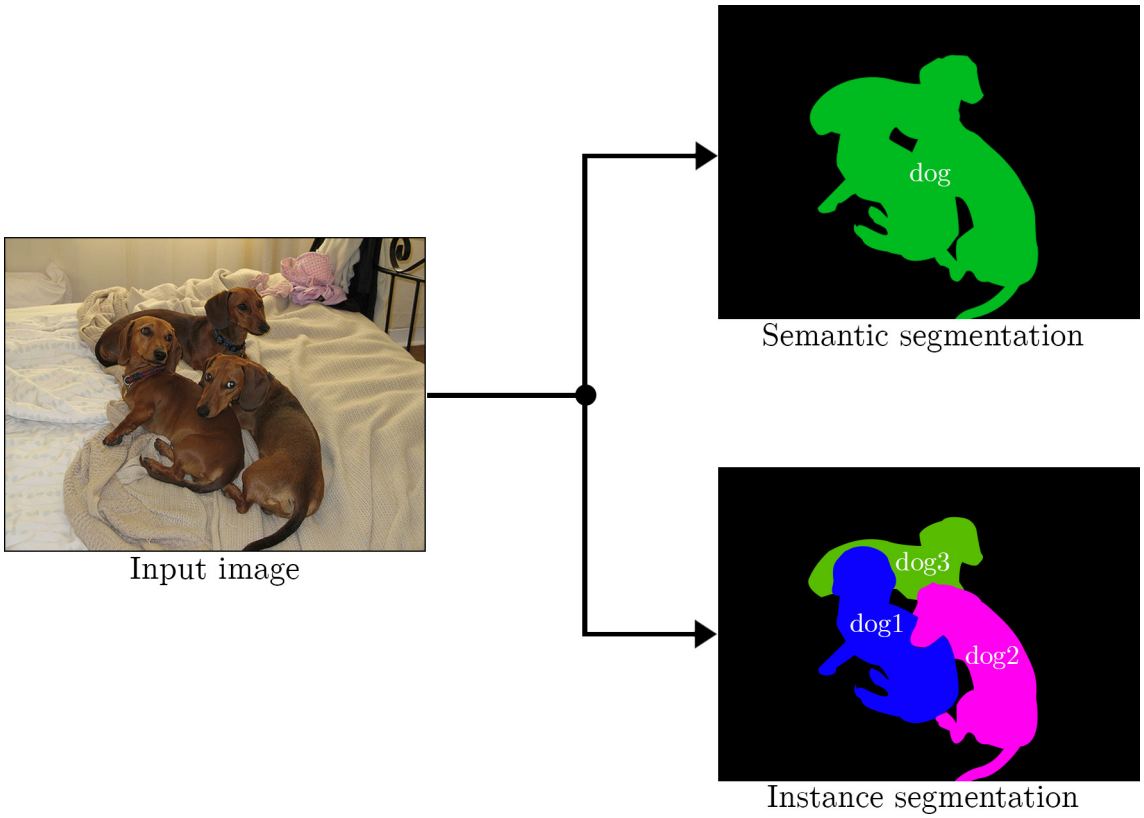


Figure 2.4: Difference between semantic segmentation and instance segmentation.

Algorithms that undertake this task need to be much more powerful than for previous image analysis tasks such as object detection and semantic segmentation. Now it becomes necessary to have a much more complete understanding of the objects in the scene and be able to, at least partially, reason about occlusion.

One of the main competition for this type of problem is the MS COCO detection 2016 competition, that provides more than 200k images and their instance-aware labels for training and evaluating performance. However, it's also common to evaluate instance segmentation algorithms using data from the Pascal VOC competition. Although its creators never established an official competition for this specific problem, it's possible to use the provided ground truth data to perform instance segmentation evaluation on this dataset.

As discussed in section 1.3, this is a very active field in the community right now, far from being solved, and with very different approaches being undertaken. At the moment there doesn't seem to be any single approach that stands out from the others, besides the fact that all the best scoring methods use a combination of CNNs or CNN-inspired architectures to accomplish this task.

3

Methodology

This chapter explains the methodology used for the project. It starts with a conceptual explanation of the main ideas of this thesis, later explaining the implementation details of them. After that, it discusses the training protocol used to optimize the networks discussed in the project and compares the different architectures proposed for the Masknet version with partial masks. Finally, it concludes with an account of how the pretraining on MS COCO data was performed for the best performing architectures of Masknet.

3.1 Conceptual explanation

Perhaps one of the most intuitive ways of looking at the problem of instance segmentation is to interpret it as trying to simultaneously solve the problems of object detection and semantic segmentation. Aided by this perspective, it's tempting to try using the key insights found when solving these problems to develop a solution to the new problem at hand.

As one can quickly realize when first confronted with the task of instance segmentation, the semantic segmentation maps produced by a semantic segmenter bear some resemblance to the desired output for the instance segmentation problem. Figure 2.4 illustrates this clearly. In the semantic segmentation map, the class-wise pixel level segmentation has already been done, the only thing missing is the separation of each one of the instances.

Looking at figure 2.2, it seems reasonable to use the output from an object detector to obtain an initial estimate of the spatial extent of each instance in the scene. A straightforward way of doing so is to, for each given class, crop the segmentation map for that class using the bounding boxes that were predicted to be from that class, thus generating as many instances as the number of found bounding boxes. Figure 3.1 illustrates this approach. As can be seen in the figure, this method is able to generate instance-wise segmentation results.

However, this approach suffers from some systematic problems. These are illustrated in figures 3.2 through 3.4. On the left part of the figures the segmentation map overlaid with the original image is shown, together with the predicted bounding boxes, while on the right the generated instance is shown. As can be seen, the quality of the output will always depend on the quality of both the predicted bounding

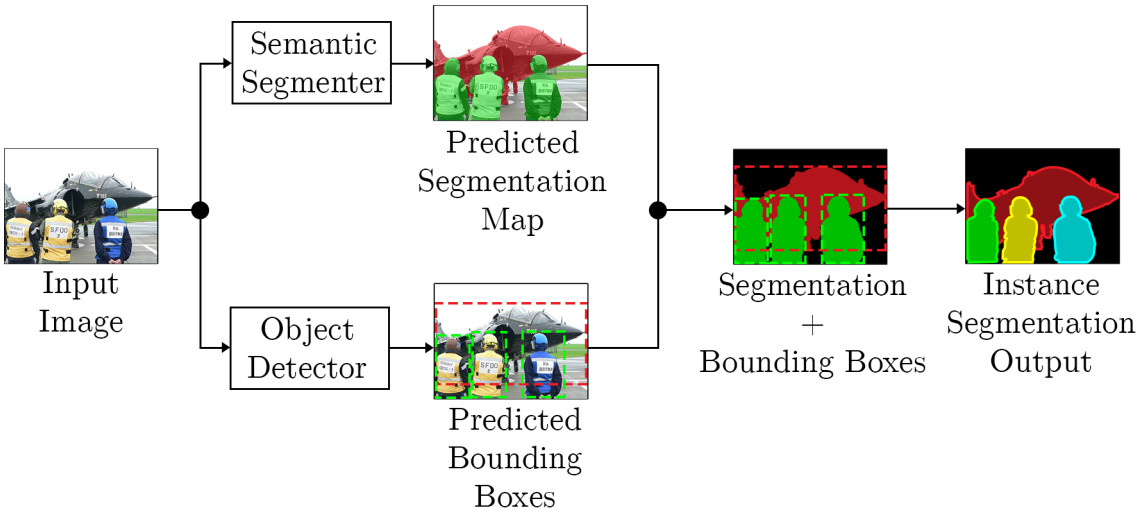


Figure 3.1: Functioning of the naive approach to instance segmentation.

boxes and the segmentation maps used. If just one of them is inaccurate, the output might suffer considerably. Additionally, even if both the segmentation maps and the bounding boxes are perfectly accurate, this method might still yield wrong predictions if any bounding boxes from the same class overlap, since this means that more than one instance might be present in the overlapping bounding box region, and the system will be unable to unfailingly disambiguate which parts belong to which instance in this overlapping region.

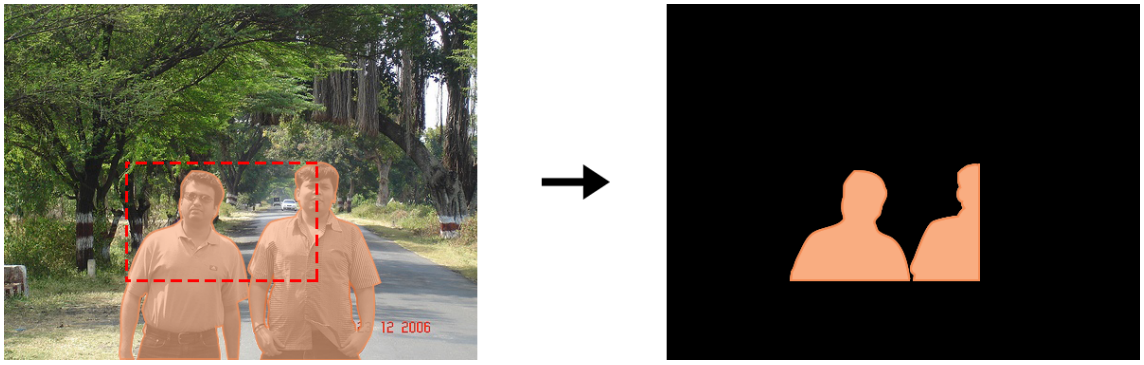


Figure 3.2: Poor instance generation due to inaccurate output from the object detector. On the left, the segmentation map for the class "person" is shown in orange, and only the bounding box for the instance being considered is shown (in red).

Given these type of errors, one possible approach could be to completely ignore the semantic segmentation maps and try predicting the binary mask for each instance directly from the visual region enclosed by each one of the found bounding boxes, thus making the system immune to errors from an imprecise semantic segmentation. Mask R-CNN [16], a recent and successful work in the field suggests that this approach is indeed worthwhile.

However, it's common that the extraneous instance parts are somewhat visually

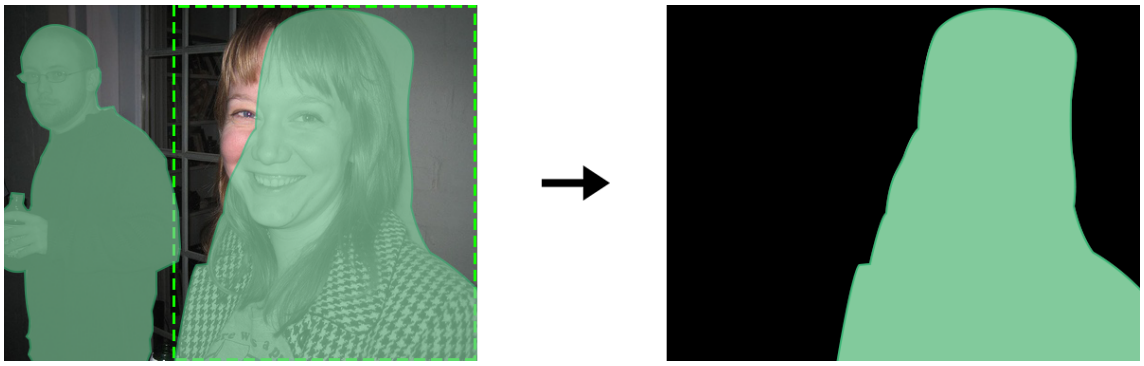


Figure 3.3: Poor instance generation due to inaccurate output from the semantic segmenter. The segmentation map for the class 'person' is shown in green, and only the bounding box for the instance being considered is shown (also in green).

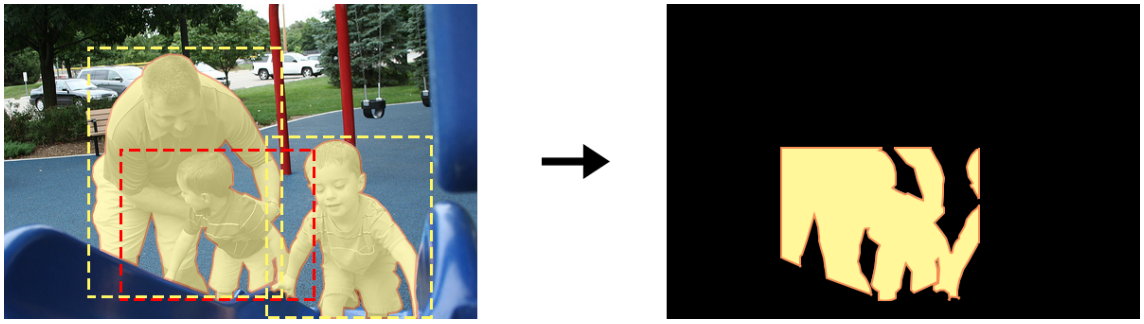


Figure 3.4: Poor instance generation due to overlapping objects. The segmentation map for the class 'person' is shown in yellow, the bounding box for the instance being considered is shown in red, and the bounding boxes for other instances are shown in yellow.

distinct and/or spatially disconnected from the majority of the instance to be segmented. Therefore, I hypothesized that it could be possible and advantageous to guide the creation of this binary mask by using an initial approximation of the pixels that are likely to belong to the instance that is to be segmented in that region. The system would take into account the parts of the instance that are already segmented to ponder whether or not a new region should be included in the binary mask.

One possible heuristic to form these type of initial partial masks is to start with the semantic segmentation map for the predicted class, cropped to the bounding box being considered, and then remove the regions that overlap with any other bounding boxes of the same class. If the bounding boxes and the semantic segmentation are precise enough, this results in a partial mask comprised of pixels that are very likely to belong to the instance, while removing most of the extraneous instances parts (clearly, at the cost of also removing perfectly fine parts of the instance to be segmented). Figure 3.5 illustrates the functioning of this heuristic.

A relevant point must be considered. In the case that two bounding boxes share the same class, and one is completely enclosed by the other, the partial mask generated for the smaller one will be empty. In these extreme cases, the system is still required

to achieve its goal. I further hypothesized that, because of the way the system is trained (detailed in section 3.3), it will still have an incentive to try to segment instances when no partial mask is provided. To some extent, this is indeed correct. Refer to section 5.1 for a more detailed description of the system's behavior in these cases.

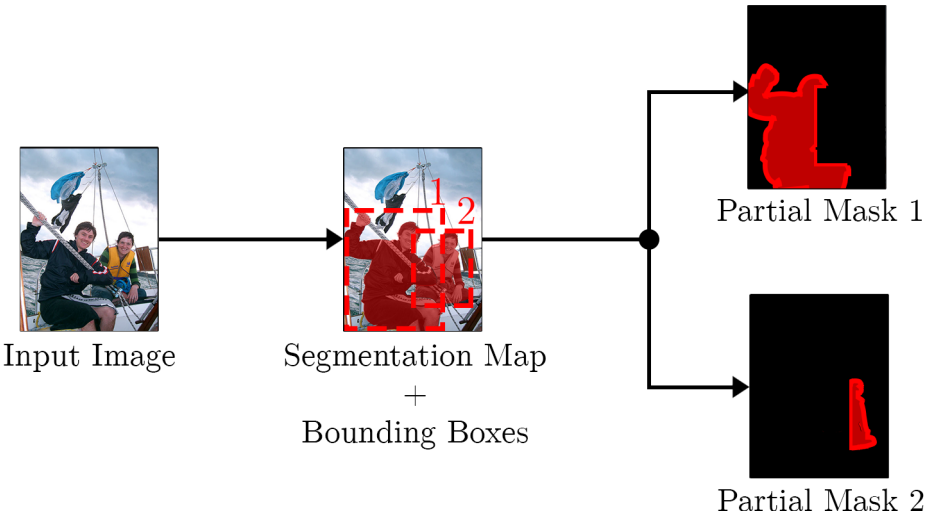


Figure 3.5: The functioning of the heuristic to create partial masks.

These two approaches (directly predicting instances from the detected object regions, or using partial masks to aid the instance generation) were implemented and evaluated. Figure 3.6 illustrates the entire system for both versions, referred to as **Masknet**. Its input is an RGB image with variable spatial dimensions, and its output is a list with all the generated instances. Each instance is comprised of a predicted class for it and a binary mask, which is a single-channel image with the same spatial size as the input, but with binary pixel values (0 denoting a pixel position which does not belong to the instance, and 1 denoting one that does).

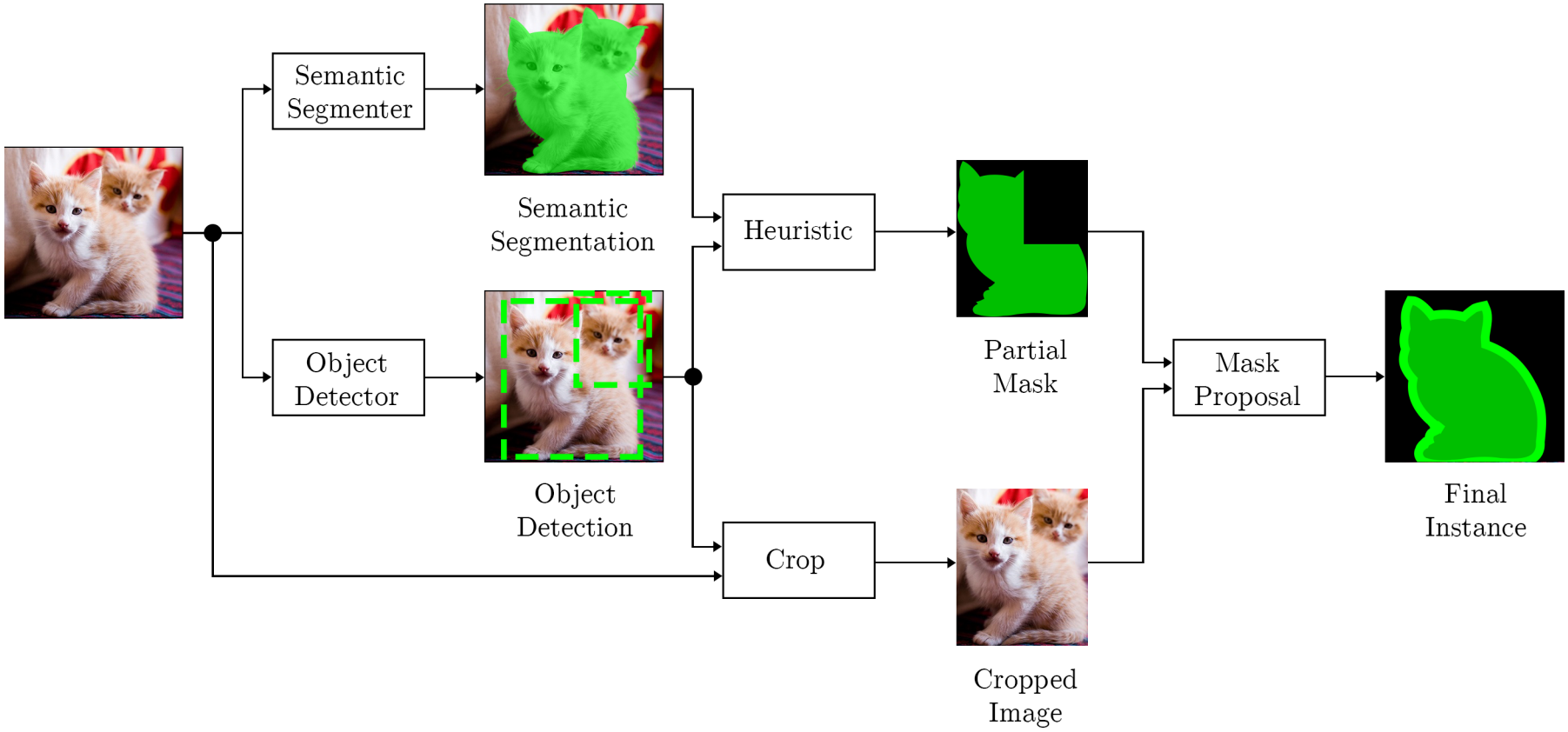


Figure 3.6: Visual depiction of the Masknet system. If the version without partial masks is being used, the semantic segmentation and the heuristic are not computed, since the "Mask proposal" subsystem does not need the partial mask as input.

3.2 Implementation

The concrete implementation of Masknet is comprised of three subsystems: an object detector, a semantic segmenter (only used by the version with partial masks) and the final mask proposal subsystem. Since one of the main focuses of this thesis is to implement and evaluate the performance and limitations of the mask proposal subsystem, the driving forces when deciding the architecture for the other parts of the project were primarily the amount of required computing power and development time. For the following networks, pretrained versions of them are available to download for the development framework used, MatConvNet, which substantially reduced the time needed to deploy them in the project.

The convolutional neural network Fast R-CNN [23] was chosen for predicting the bounding boxes for each object in the scene, whereas FCN-8s [24] was chosen for the semantic segmentation task. These networks achieve close to state-of-the-art performance in their respective tasks for both datasets used, while also remaining within the boundaries of available resources for this project.

For the mask proposal system, a new architecture was designed, inspired by the success and design choices from DeepMask [25]. The part of their architecture for predicting instance masks is illustrated in figure 3.7.

The approach is to first bilinearly resample the input RGB image to a spatial size of 224x224, then the network extracts features from the input using the CNN VGG-16 with all the fully connected layers as well as the last max-pooling layer removed, and exchanged by a convolution with a filter bank of size 1x1x512, followed by a ReLU layer. This results in 512 feature maps of spatial size 14x14.

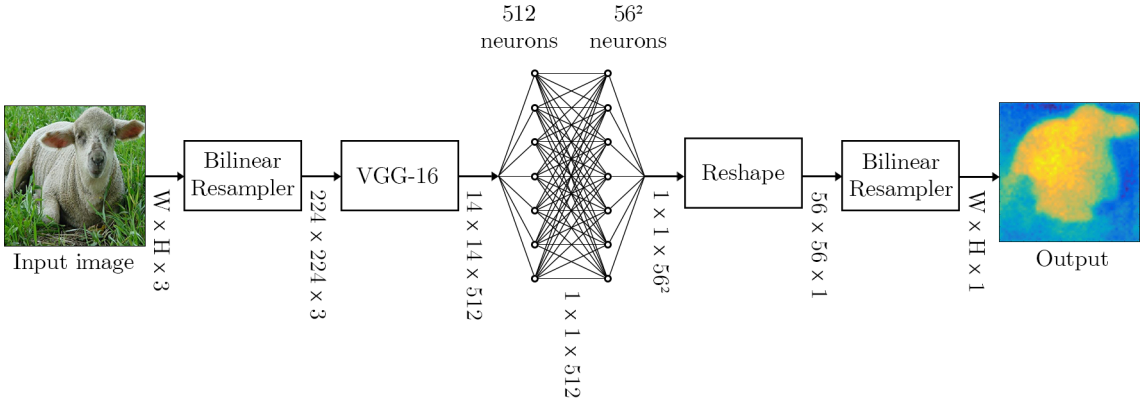


Figure 3.7: DeepMask architecture. "VGG-16" is the altered version of the VGG-16 CNN, as mentioned in the text. The vertical text between each layer demonstrates the size of the data in that step, and the "Output" subimage is a heatmap illustrating the mask scores (yellow means high probability of being part of the mask, while blue means low).

The extracted features maps are then passed through a fully connected stage that outputs a vector with 56^2 dimensions. To reduce the number of learnable parameters in the network, instead of using a single fully connected layer to perform this step,

the authors of DeepMask decompose this into two stages. First the feature maps are connected to 512 neurons, and then each one of these neurons is connected to all entries of the output vector.

This vector is in turn reshaped into a 56x56 single channel image, bilinearly resampled to the original input size and optionally thresholded to yield a binary mask. This is exactly how the mask proposal subsystem for the Masknet version without partial masks was implemented. However, for the other version of Masknet, it's desired to use the information from the partial masks to guide the creation of the final mask. For this, three different merging strategies were proposed and evaluated. Each one of these is explained below, together with the necessary modifications to the overall structure in each case.

1. The partial mask is bilinearly resampled to a spatial size of 14×14 and concatenated to the feature maps as a new feature. The first fully connected layer is then adjusted to connect the 513 feature maps to each of the subsequent 512 neurons.

2. Same as the above, but after concatenating, and before the first fully connected layer, a filter bank of size 1×1×513×512 (i.e. 512 filters of spatial size 1x1 and 513 channels) convolves the new feature maps, followed by a ReLU layer. As a result, there's no need to change the rest of the network (since the convolution results in a data object of size 1×1×512, the same size expected by the original DeepMask architecture).

3. Fully connect the entire feature map *and* the partial mask in its original spatial size to the the output vector, in a single (potentially very big) fully connected layer.

   To ensure that the fully connected layer's size stays within reasonable bounds, a filter bank of size 14×14×512×N — where N is a positive integer in the range 100 to 512 — is added right after the feature extractor, in order to select a smaller linear combination of the 512 VGG features. In this way, the parameter N generates a family of closely related merging strategies with a different compromise between the number of features used (which is likely to correlate with performance) and required computing power. This is then followed by a ReLU layer.

The different merging strategies are illustrated in figures 3.8 through 3.10. The parts of the diagram after the fully connected layer are the same as in the original DeepMask architecture, and therefore are not shown. Section 3.4 explains how one of these merging strategies was chosen for the final architecture.

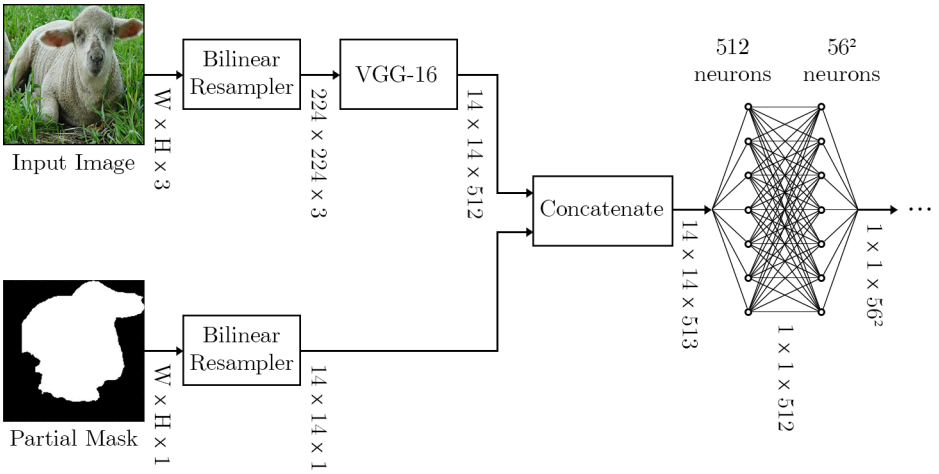


Figure 3.8: Merging strategy 1. The partial mask is bilinearly resampled to the same spatial size as the feature maps, and then concatenated to the end of it as a new feature.

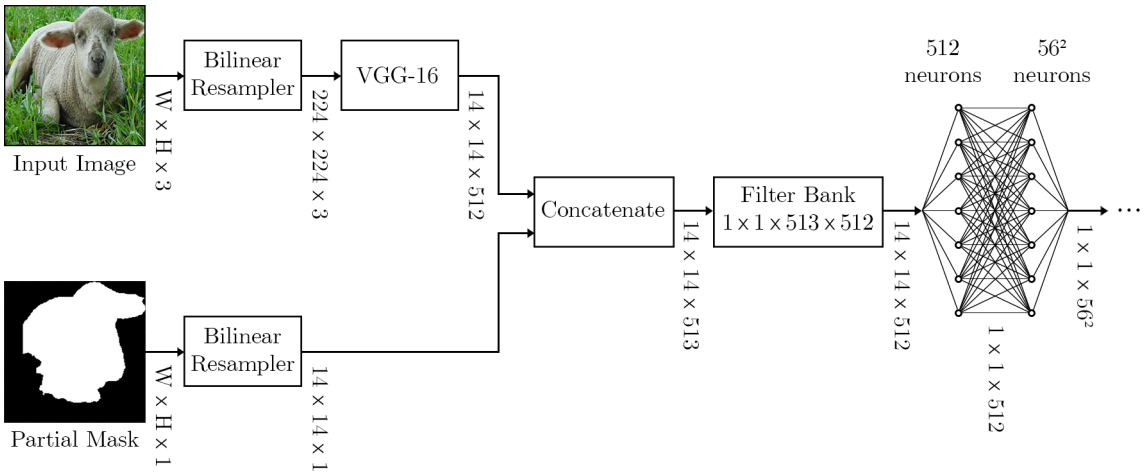


Figure 3.9: Merging strategy 2. After the concatenation, the new feature maps are convolved with a filter bank that transforms it back to the size expected by the rest of the network (followed by a ReLU layer, not shown in the picture).

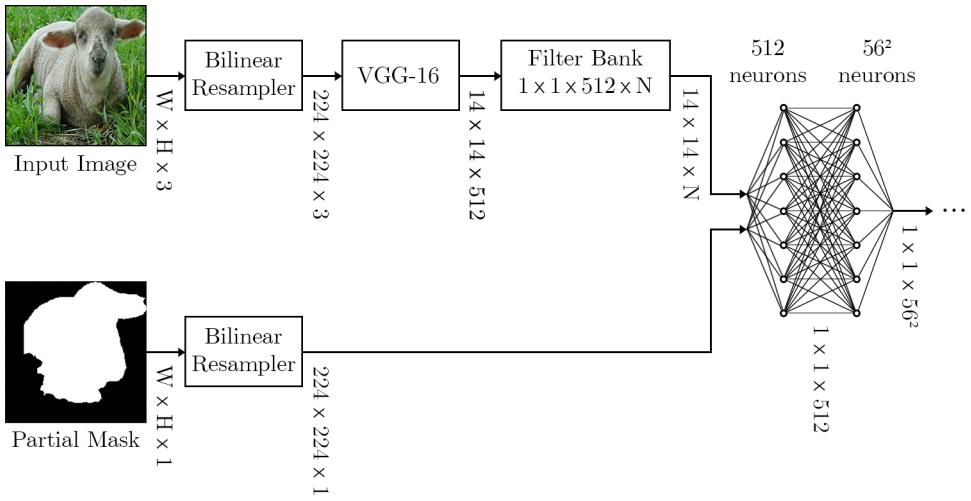


Figure 3.10: Merging strategy 3. Instead of resampling the partial mask, it is directly connected to each neuron in the fully connected layer, together with a subset of the feature maps from VGG-16.

3.3 Training protocol

This section describes the specific design decisions made for training the Masknet system. First of all, it's important to note that the only part of the system being trained is the mask proposal subsystem. For the semantic segmenter and the object detector, the pretrained versions of FCN-8s and Fast R-CNN are used, respectively.

Since the pretrained networks are being used, it's important to use the same dataset in which they were pretrained on, in order for them to achieve their best performance. In this case, it's the Pascal VOC dataset. This dataset has several different versions, one for each competition held by the authors. Statistics summarizing three relevant versions of this dataset are presented in table 3.1.

Table 3.1: Number of images and objects present in different versions of the Pascal VOC dataset.

	Training set		**Validation set**	
	Images	**Objects**	**Images**	**Objects**
Pascal VOC 2007	209	633	213	582
Pascal VOC 2011	1112	2501	1111	2533
Pascal VOC 2012	1464	3507	1449	3442

Since FCN-8s was pretrained in Pascal VOC 2011, and Fast R-CNN in Pascal VOC 2007, there is no single version of the Pascal VOC datasets that would be best for performing the training for the mask proposal subsystem. Choosing the Pascal VOC 2007 dataset would likely yield the best performance for both the semantic segmenter and the object detector (since the object detector was trained precisely

in this version of the dataset, and the later versions of Pascal are supersets of the older ones). However, as can be seen in table 3.1, this would also provide far less examples for the training.

With this is mind, a visual assessment of the performance of these pretrained systems on the Pascal VOC 2012 (latest) validation dataset was carried out. This showed that these systems still achieve an acceptable performance on this dataset version. Based on this visual assessment, I chose the Pascal VOC 2012 dataset for the training. Although the pretrained networks might suffer in performance, for the purposes of generating the training examples for the mask proposal subsystem, a much larger number of examples is preferred over slightly more accurate ones (specially since ideally the mask proposal subsystem should be reasonably unaffected by inaccurate inputs). A better approach, unfortunately left as a proposal for further work due to limited time constraints, would be to retrain the FCN-8s and Fast R-CNN networks with the training data from Pascal VOC 2012 or, even better, with the data from MS COCO 2016. This approach is further detailed in section 5.3.

Each image in the Pascal VOC dataset is processed by the object detector, the semantic segmenter and the heuristic to generate partial masks. This generates as many training examples as the number of detected objects in the scene. Each of the generated training examples is comprised of an input image, the partial mask, and the ground truth for that example. The final output from the mask proposal subsystem is compared to the ground truth by using the pixel-wise logistic log loss function, shown in equations 3.1 and 3.2:

$$L(x, c) = \frac{\sum_{j=1}^{B} \sum_{i=1}^{N} M(x_i^j, c_i^j)}{B} \tag{3.1}$$

$$M(x_i^j, c_i^j) = \log(1 + e^{-c_i^j \ x_i^j}) \ , \tag{3.2}$$

where x is the $W \times H \times B$ matrix with all the predictions in the current batch, W and H denote the spatial size of the predictions, B is the number of images in the current batch), c is the $W \times H \times B$ matrix with all the labels for the images in the current batch, x_i^j is the i-th pixel of the prediction for the j-th image, c_i^j is the i-th pixel of the label for the j-th image (which is 1 if the pixel should be in the mask, and -1 otherwise), and N is the number of pixels in the image (namely, $W \times H$).

The value computed from this loss function is then backpropagated through the mask proposal subsystem network for parameter learning. For this learning task, gradient descent with momentum and weight decay was chosen as the optimization algorithm.

For a more humanly understandable assessment of the performance, the output mask quality is also evaluated using the metric of IoU (Intersection over Union). The output mask is binarized, and the IoU with ground truth is computed by dividing the intersection (number of pixel locations with the value 1 in *both* the binary mask and the ground truth) by the union (number of pixel locations with the value 1 in *at least one of* the binary mask and the ground truth) of them. The intersection and union values are aggregated for all images in the batch before the division.

3.4 Evaluation of different merging strategies

In order to decide which of the merging strategies defined in section 3.2 should be used, a preliminary evaluation of the mask proposal subsystem was carried out for each one of them.

The training protocol specified in section 3.3 was followed and each one of the three merging strategies was trained and evaluated using the data from Pascal VOC 2012. The network weights were initialized randomly, using Xavier initialization [26]. The hyperparameters for each strategy were chosen according to the following steps:

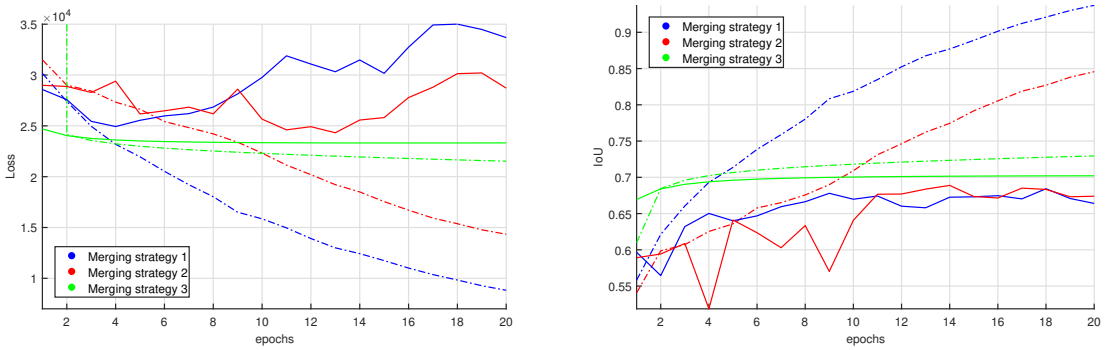
1. Choose a batch size so that each batch fits in the available memory.
2. Run the optimization on Pascal VOC data with no momentum, no weight decay, and different values of learning rate (10 logarithmically spaced values between 10^{-7} to 10^{-4} — no strategy converges with a learning rate higher than 10^{-4}).
3. Choose the learning rate that yielded the second best training error after 10 epochs of training (the best performing one might have convergence problems in the long run).
4. Run the optimization using the chosen learning rate for each one of the following hyperparameter values
   (a) Momentum = 0, Weight decay = 0
   (b) Momentum = 0.9, Weight decay= 0
   (c) Momentum = 0, Weight decay = 5×10^{-5}
   (d) Momentum = 0.9, Weight decay = 5×10^{-5}
5. Choose the combination of hyperparameter values from step 4 that yields the lowest validation error.

Since this was only a preliminary evaluation, no effort was spent in searching the hyperparameter space further. Using the aforementioned steps, the optimal hyperparameters found for each merging strategy are summarized in table 3.2.

Table 3.2: Hyperparameters found for each one of the merging strategies.

Merging strategy	Learning Rate	Momentum	Weight Decay
1	2.1544×10^{-6}	0.9	5×10^{-5}
2	4.6416×10^{-6}	0.9	5×10^{-5}
3	4.6416×10^{-5}	0	0

The training and validation errors for the optimization using these values are shown in figure 3.11. For merging strategy 3, several values of N were tested (the size of the filter bank that convolves the VGG features into a smaller subset), and the best performing one within the hardware resources available, $N = 300$, is the one shown in the plots.



(a) Loss value. **(b)** IoU value.

Figure 3.11: Evaluation results for the three merging strategies. Each color specifies a different strategy, and the dashed and solid lines denote if the value is computed in the training set or the validation set, respectively.

As the figure demonstrates, all merging strategies achieve a reasonable IoU in the validation set, but merging strategy 3 outperforms the other two considerably. Therefore, this was the merging strategy chosen for the final implementation of the system and for the comparison with the version that does not use partial masks.

3.5 Pretraining on MS COCO

In order to improve results even further, both the version that does not use partial masks and the version that best uses them (i.e. merging strategy 3) were retrained on Pascal VOC. However, this time the hyperparameters were optimized by trial-and-error (starting at the values found earlier) and the networks weights were not initialized randomly. Instead, these two versions were first trained on MS COCO data for 30 epochs, and the network weights at the best performing epoch were used to initialize the training on Pascal VOC 2012. This section describes how this pretraining on MS COCO was performed.

As explained in section 3.2, the versions of Fast R-CNN and FCN-8s used in this project were trained with data from the Pascal VOC dataset (Pascal VOC 2007 for fast R-CNN and Pascal VOC 2011 for FCN-8s). Retraining both of these networks in the MS COCO 2016 dataset would require a considerable amount of time and computing power. Instead, ground truth data from this dataset is used to simulate a perfect object detector and a perfect semantic segmenter. To do so, each image in the dataset is processed before being fed to the system. The procedure is illustrated in figure 3.12.

For each object present in the ground truth annotation for the image, a training example comprised of input image, partial mask and ground truth is generated with the following steps. First the original image is cropped to the bounding box of the current object, generating the input image. Then, using the ground truth segmentation map and the ground truth bounding boxes, the partial mask is generated using the heuristic discussed in section 3.1. Finally, the binary mask for the current object,

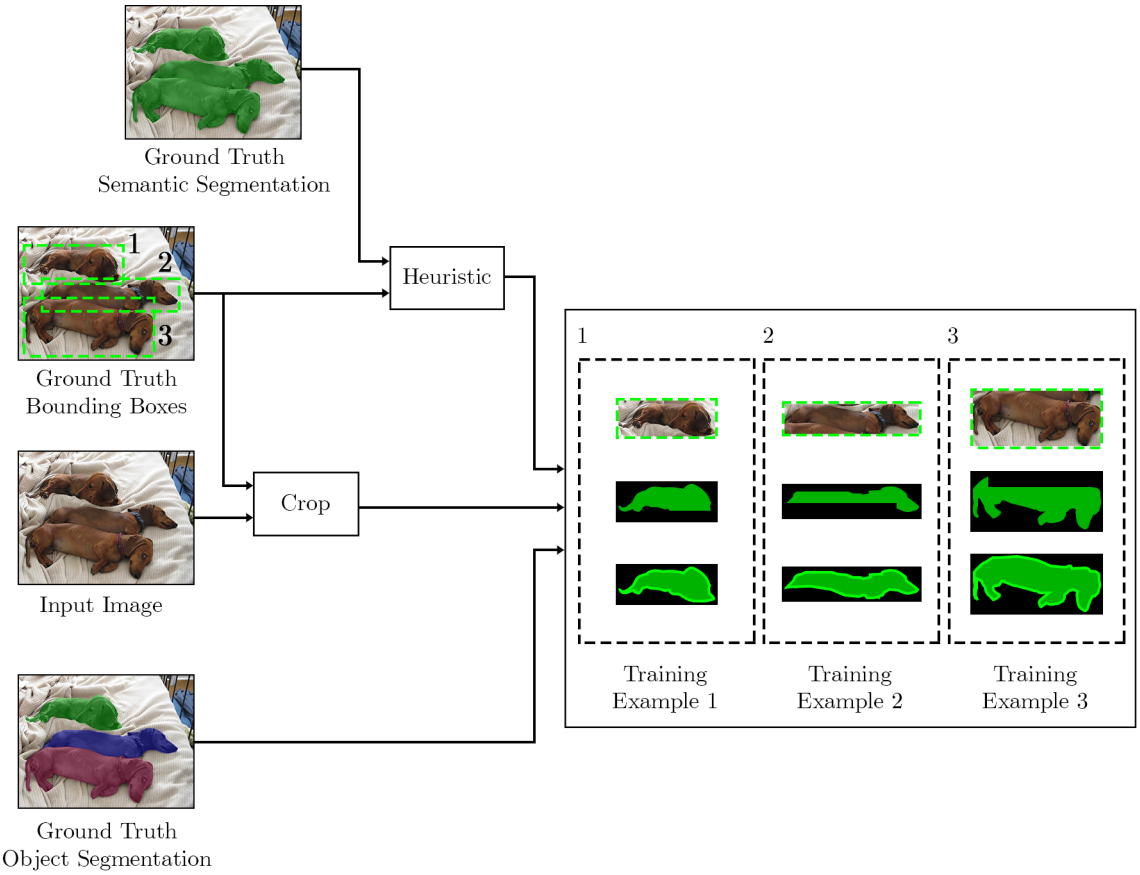


Figure 3.12: Procedure for generating training examples for the preliminary evaluation of the merging strategies, using ground truth data from MS COCO.

obtained from the ground truth object segmentation, is saved as the ground truth for this training example. This procedure generates as many training examples as there are ground truth object annotations in each image.

The motivation behind this pretraining is that MS COCO has a much bigger and more varied collection of images (around 300k images in MS COCO, compared to 3k in Pascal VOC 2012). Hence, using the weights from a network trained in MS COCO (even if ground truth data was used, instead of real data from an object detector and a semantic segmenter) should provide a much better starting point than random initialization when training with Pascal VOC data.

Because of the way the chosen heuristic was designed, and the fact that we are using ground truth data to generate the partial masks, objects that have bounding boxes that do not overlap with other bounding boxes of the same class would have a perfect partial mask (i.e. identical to the ground truth mask for that object). To avoid such artificial examples, the only objects kept are those with bounding boxes that overlap any other bounding box of the same class by more than 10%.

70% of all the kept examples are used for training, while 30% are held separate as the validation set. During training, each of the examples in the training set are fed into the system, and the output mask is evaluated by comparing it to the ground

truth using the same logistic log loss function described in section 3.3. The value from this loss function is then back-propagated through the network for parameter learning. Additionally, just as described in section 3.3, the output mask quality is also evaluated using the metric of IoU, and the epochs with the best scoring IoU in the validation set are the ones chosen for further analysis, described in section 4.1.

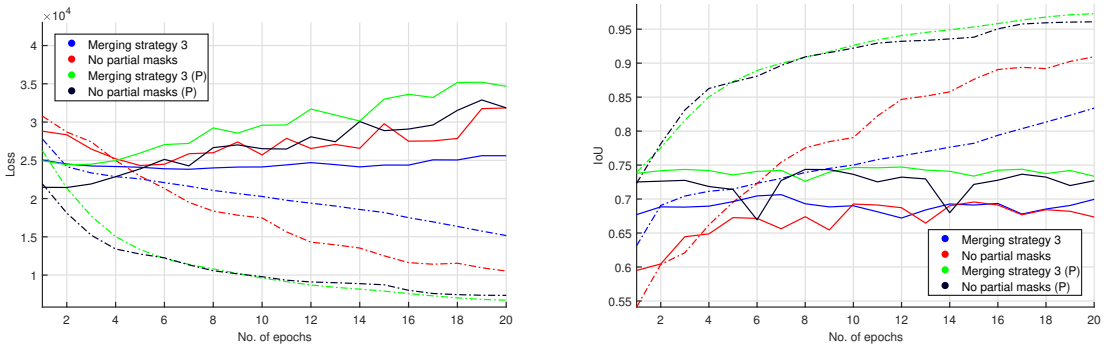
4

Results

This chapter presents the results of training the two versions of Masknet, the one with partial masks (using merging strategy 3), and the one without. Furthermore, it also presents a subset of the precision-recall curves for both versions, the average precision for each class, and the MAP^r metric value for them, together with a brief interpretation of these results, ending with example outputs from the system.

4.1 Training results

After the preinitalization on MS COCO, the training protocol from section 3.3 was followed. The objective function value and the IoU, computed in both the training and validation set, are shown in figure 4.1 for the relevant versions of Masknet, given a training length of 30 epochs. The hyperparameters used for this training are specified in table 4.1.



(a) Loss value.

(b) IoU value.

Figure 4.1: Training results for both versions of Masknet and their preinitialized versions. The dashed lines depict the error on the training set and the solid lines on the validation set. The versions that used preinitialized weights are shown with a (P) in their legends.

As can be seen in the figure, preinitializing the networks improved the optimization substantially, for both the version that uses partial masks and the one that does not. Further, the corresponding best IoU for both preinitialized versions is only slightly different (0.5%), so no conclusions can be drawn as to which version will perform

Table 4.1: Hyperparameters used in the training.

Architecture	Learning Rate	Momentum	Weight Decay
Merging strategy 3	2×10^{-6}	0.9	0
No partial masks	1×10^{-6}	0.9	5×10^{-5}
Merging strategy 3 (P)	1×10^{-6}	0.9	0
No partial masks (P)	1×10^{-6}	0.9	5×10^{-5}

better. Subsection 4.2.2 assesses the performance of the entire system (not just the mask proposal), and there more substantial differences can be found.

4.2 Performance evaluation

The final training and validation errors for the optimization of the mask proposal subsystem can only quantify how well this particular subsystem is performing its task. To evaluate the performance of the system as a whole, a specific metric for the problem of instance segmentation is used, called the Mean Average Precision (abbreviated as MAP^r).

4.2.1 The MAP^r metric

First introduced in [5], this metric is an attempt to quantify how well a system is performing the task of instance segmentation based on its precision-recall curves for each class. First, the precision-recall curves are obtained for the algorithm being evaluated. Then, the area under each curve is computed, each one being denoted as the average precision for that specific class. Finally, the MAP^r metric is defined as the average of average precisions over all predefined classes for the problem — hence the name **Mean** Average Precision (the superscript ^{r} in its abbreviation is used to differentiate this metric from a similar one defined for the problem of object detection, denoted MAP^b).

To generate the precision-recall curves for an algorithm, first it's necessary to match the generated instances with the ground truth annotated objects present in each one of the images. A generated instance matches with a ground truth object if both share the same class and if their IoU is greater than a predefined threshold. If an instance matches several ground truth objects, only the highest scoring IoU match is kept.

Higher thresholds render stricter MAP^r metrics, as different applications may require. To clearly specify which IoU threshold is being used, it's common to use the notation MAP^r@X, where X is the threshold value used when computing the metric.

Once a generated instance is matched with a ground truth object, that object is removed from consideration and cannot be matched to any other generated instances, thus penalizing repeated instances (which will have no matches, and therefore be

counted as false positives, as explained below).

After the matches are found for all images, precision and recall are calculated. Precision is a measure of how many instances, among the predicted, are correct; whereas recall measures how many instances, among all ground truth ones, where correctly found. These are computed as

$$P = \frac{tp}{tp + fp} \tag{4.1}$$

$$R = \frac{tp}{tp + fn} \ , \tag{4.2}$$

where P is the precision, R is the recall, tp is the number of true positives (instances with a matching ground truth object), fp is the number of false positives (instances with no matching ground truth object) and fn is the number of false negatives (ground truth objects with no matching instances). tp, fp and fn are aggregated for all images.

Finally, for a precision-recall curve to be assembled, it's necessary for the instance segmentation algorithm to rank the instances generated with a confidence level between 0 and 1. Then, precision and recall are calculated as described above, but using only instances with a confidence level higher than a threshold. By gradually changing this threshold from 0 to 1, a series of precision-recall pairs are generated, and can then be plotted in a precision-recall plane for that class.

To rank the generated instances as needed, the confidence level assigned to each instance was the confidence level assigned to its bounding box by the object detector. Although this was the only ranking strategy used, other strategies, perhaps aided by the mask proposal subsystem, might lead to a better performance, as discussed in section 5.3.

Once the precision-recall points are generated, the average precision for a given class can be approximated as

$$AP = \sum_{n=1}^{N} [R(n) - R(n-1)] \cdot \max_{\tilde{n} \geq n} P(\tilde{n}) \tag{4.3}$$

where AP is the average precision, $P(n)$ and $R(n)$ are respectively the precision and recall of the point with the n'th lowest recall, and N is the number of precision-recall points generated. If there are several points with the same recall value, all but the one with the highest precision are discarded. The average precision approximated this way is denoted interpolated average precision [27].

One way to get some intuition about what this equation is doing, is to note that, if the precision-recall curve is monotonically decreasing, this approximates the area under it using rectangles of height $P(\tilde{n})$ and width $[R(n) - R(n-1)]$. If the curve is not monotonically decreasing, then the max operator approximates that curve with one that is and has the smallest possible integral in that interval.

4.2.2 Instance segmentation results

Using only data from Pascal's validation set (there is no public ground truth for the test set), the precision-recall curves for Masknet were computed as described in the previous section, using an IoU threshold of 0.5 (commonly used for the Pascal VOC datasets), and a subset of them are shown in figure 4.2. The average precisions for each class were computed as described in section 4.2.1, and are shown in tables 4.2 and 4.3.

Table 4.2: AP values for the first 10 Pascal VOC classes, calculated with IoU threshold = 0.5.

Version	Aeroplane	Bicycle	Bird	Boat	Bottle	Bus	Car	Cat	Chair	Cow
With partial masks	0.70	0.06	0.72	0.41	0.33	0.76	0.56	0.85	0.16	0.67
No partial masks	0.74	0.04	0.70	0.48	0.35	0.78	0.61	0.82	0.17	0.66

Table 4.3: AP values for the last 10 Pascal VOC classes, calculated with IoU threshold = 0.5.

Version	Dining Table	Dog	Horse	Motorbike	Person	Potted Plant	Sheep	Sofa	Train	T.V. Monitor
With partial masks	0.30	0.84	0.57	0.70	0.56	0.35	0.59	0.44	0.75	0.70
No partial masks	0.36	0.85	0.62	0.74	0.60	0.42	0.59	0.51	0.82	0.69

These results show that both systems are able to perform the task of instance segmentation to a reasonable standard, but performance varies considerably between different classes. For instance, both versions have very high average precisions for the dog and cat classes, which have their precision-recall curves illustrated in figure 4.2a and 4.2b. However, both also have a lot of difficulty with the bicycle and chair classes, which have their precision-recall curves illustrated in figure 4.2c and 4.2d. To some degree, a bad performance in these two classes was already expected, since they are indeed among the most difficult ones in the dataset.

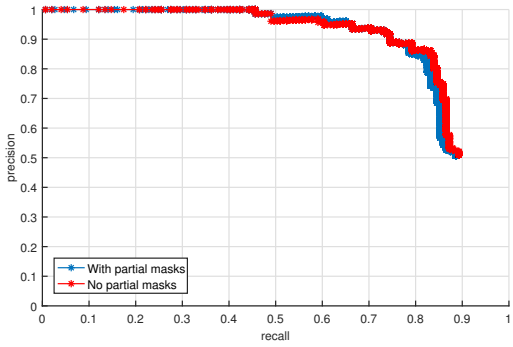
Bicycles have a very strict ground truth mask, where only the pixels that exactly belong to the instance are accepted. For example, all the pixels in a bicycle wheel are accepted as correct, and also the spikes, but not the space between the spikes (in some cases, correctly separating these might be difficult even for a human). Chairs very often appear partially occluded in the images, making it also harder for algorithms to correctly identify and segment instances.

For some of the classes, like the sheep class for instance (precision-recall curve shown in figure 4.2e), there is not much difference between the two different versions of Masknet. On the other hand, there are classes where the version with partial masks is the best (e.g. cat class), and there are some in which the opposite is true (e.g. boat class, which has its precision-recall curve shown in figure 4.2f). To obtain a wider view about the performance of each version, their mean average precisions were computed across all classes (i.e. their MAP^r metric), and the results are shown in table 4.4. For a more conclusive assessment of performance, the MAP^r of the different versions was computed using two different IoU thresholds, 0.5 and 0.7, and the results for other state of the art instance segmentation algorithms are also shown.

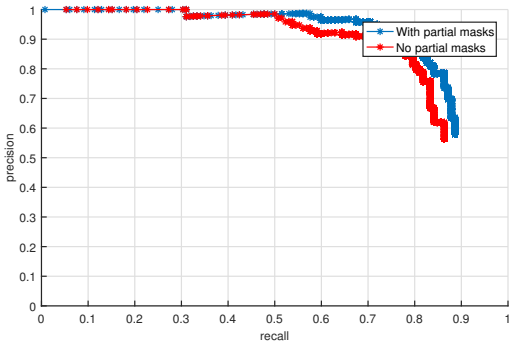
Although unexpected, since the version with partial masks requires much more computing power, the version without partial masks actually achieves the best performance in both IoU threshold levels. Section 5.1 investigates possible explanations for this. Using the version without partial masks, Masknet obtains competitive results. Furthermore, some of the improvements discussed in section 5.3 are reasonably straightforward (such as training the object detector on Pascal VOC 2012, instead of 2007) and have the potential to improve the results considerably. Using the best version, the output for some of the images from the validation set is shown in figure 4.3.

Table 4.4: MAP^r scores for Masknet and different algorithms. MNC was the winner of the MS COCO 2015 detection competition, and FCIS of the 2016 competition.

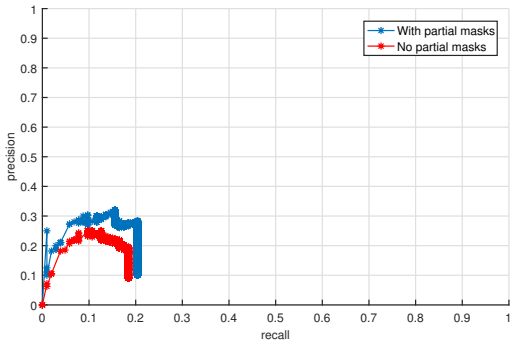
Method	**MAP^r@0.5 (%)**	**MAP^r@0.7 (%)**
O^2P [28]	25.2	-
MultiInstanceObjectSeg [29]	46.3	27.0
SDS (AlexNet) [5]	49.7	25.3
Masknet (with partial masks)	**55.1**	**34.2**
Masknet (no partial masks)	**57.7**	**35.2**
Hypercolumn [8]	60.0	40.4
CFM [30]	60.7	39.6
MNC [9]	63.5	41.5
BAIS [14]	65.7	48.3
FCIS [10]	65.7	52.1



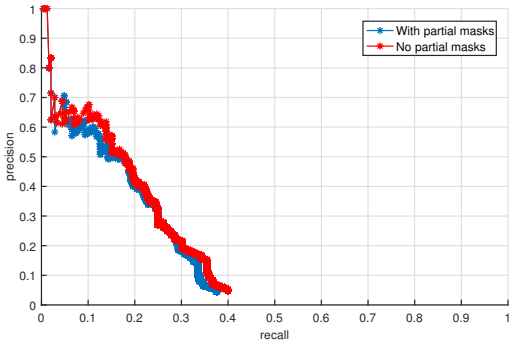
(a) Dog class.



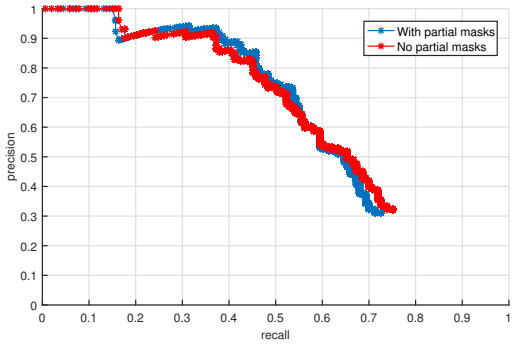
(b) Cat class.



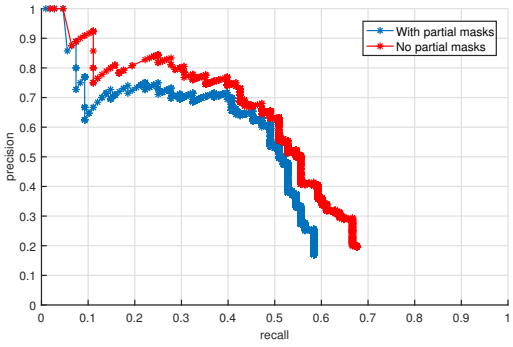
(c) Bicycle class.



(d) Chair class.



(e) Sheep class.



(f) Boat class.

Figure 4.2: Precision-recall curves for the two different versions of Masknet, for six different classes.

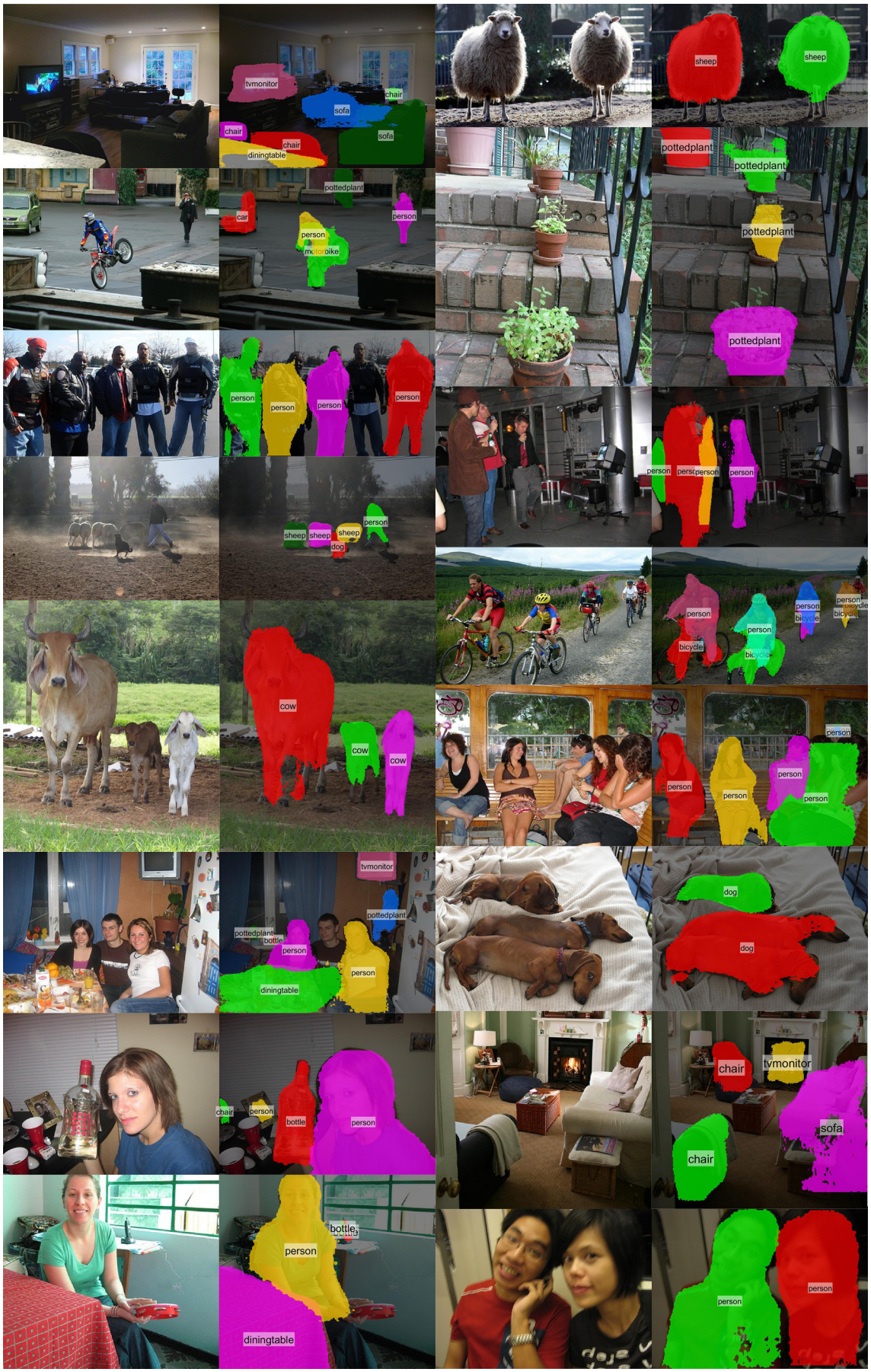


Figure 4.3: Masknet's output for images in the validation set.

5

Discussion

This chapter provides a more in-depth evaluation of the algorithm, presenting a visual assessment of its outputs, which illustrates some success and failure cases. After that, a section devoted to evaluating the impact of each of the Masknet's subsystems to the final instance segmentation quality is presented. Finally, this chapter ends by providing a section with proposals for further work that have the potential to improve the Masknet system.

5.1 Visual assessment of the mask proposal subsystem's performance

To try to gain some insight into why the Masknet version with partial masks does not outperform the other version, the input and output from the mask proposal subsystem for this version of Masknet were visually analyzed, and displayed here. All the images in this subsection are taken from the validation set, since, as mentioned before, there is no public ground truth available for Pascal VOC's test set.

First, as expected, the system is indeed capable of reconstructing the missing parts that are visually similar to the instance being considered, as one can see in figure 5.1. In each example, the image on the left is the patch that was input to the mask proposal subsystem, the center image is the partial mask for that example, and the image on the right is the output. The front of the train, the legs of the man walking, and the tail of the sheep, although not present in the corresponding partial masks, were correctly reconstructed in the output.

Furthermore, the system is also able to reconstruct missing parts that do not possess a strong visual similarity to the rest of the partial mask, as can be seen in figure 5.2. Both the bus windshield and the television frame are not visually similar to the majority of the provided partial mask.

Additionally, the system can also remove extraneous pixels that are present in the partial mask, but are not part of the instance being considered, as shown in figure 5.3. A substantial amount of the image around the body of the woman, and between her legs, was classified as belonging to the instance, according to the partial mask, but the system correctly rejected these extraneous pixels in its output (at the worthwhile cost of removing some correct pixels as well).

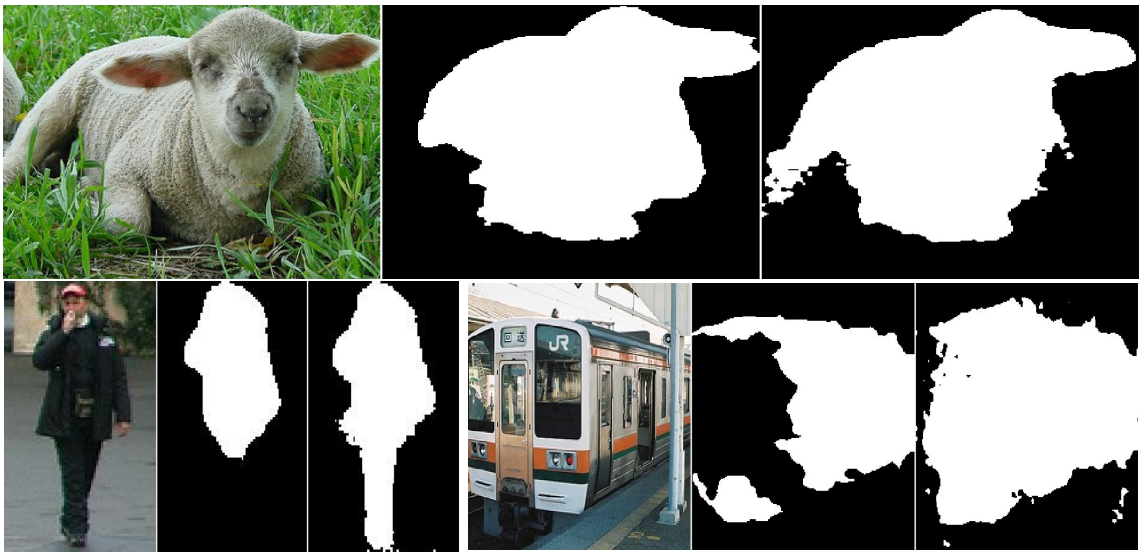


Figure 5.1: Images in which the Masknet system correctly reconstructed missing instance parts that have strong visual similarity to the provided partial mask.

Figure 5.2: Images in which the Masknet system correctly reconstructed missing instance parts that do not posses any strong visual similarity to the provided partial mask.

Figure 5.3: Images in which the Masknet system correctly removed extraneous instance parts from the provided partial mask.

However, the system is not able to recover from strong mislocalizations (i.e. very imprecise predicted bounding boxes), as shown in figure 5.4. In the cases illustrated, just one bounding box was predicted for both instances. Hence, both the image and the partial mask inputs contain the two instances, and the mask proposal subsystem

is not able to correctly segment just one of them.

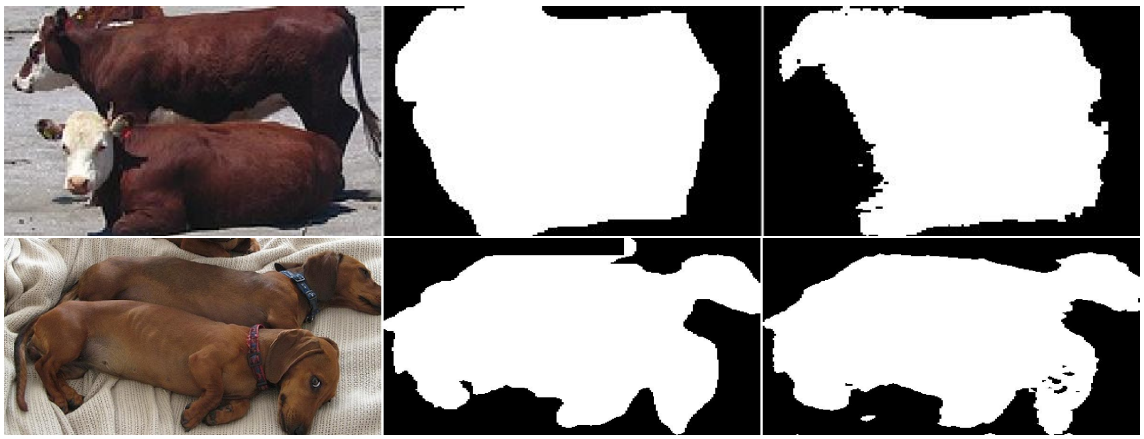


Figure 5.4: Images in which the Masknet system was unable to recover from a strong mislocalization.

If instead the bounding boxes have a reasonably good quality, the system is indeed capable of separating individual instances, as shown in figure 5.5. Even though the body of another dog is also present in the input image, good bounding boxes were predicted for each object in the scene, resulting in a partial mask that does not contain the extraneous dog, and a mask proposal output that correctly identified just the instance being considered.

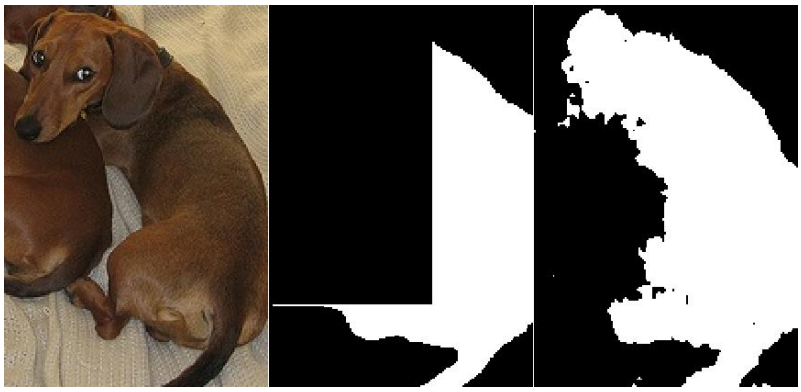


Figure 5.5: Image in which the Masknet system was able to correctly select the desired instance, even though extraneous parts that possess a strong visual similarity are also present in the image.

Also, even in the extreme cases that the designed heuristic generates completely empty partial masks, the system is still able to reconstruct the instance to some extent, as shown in figure 5.6.

Unfortunately, there are several examples in which this version of Masknet outputs bad instance masks, even with good predicted bounding boxes and a good partial mask, as shown in figure 5.7.

All of these examples demonstrate that this version of the system is indeed capable of performing the task of instance segmentation to some extent, suggesting that

Figure 5.6: Images in which the Masknet system was able to reconstruct the desired instance to some extent, even though the provided partial mask is completely empty.

Figure 5.7: Images in which the Masknet system was unable to correctly segment the desired instance, even though good bounding boxes and a good partial mask were obtained.

using partial masks could indeed be beneficial. However, no particular pattern in the output could be associated with the failure cases presented (besides strong mislocalizations).

One possible explanation for the fact that this — much more computationally demanding — version of Masknet does not outperform its simpler version, is that the architecture chosen for this version is not the optimal one, and there could be another sequence of layers (for instance, not based on DeepMask), or another merging strategy that performs instance segmentation much better, therefore justifying the higher computational complexity of this version.

Although it's not possible to rule out this theory, I conjecture that the architecture and merging strategies proposed, although not necessarily optimal, should have enough power to outperform at least the version that does not uses the extra information contained in the partial masks. Further, this high capacity of the proposed architecture is where I suspect the problem might come from: overfitting.

In the training results for Masknet, shown in figure 4.1, one can see that, when the validation IoU attains its maximum value for each version, comparing the training values with the validation values shows that there is a considerable amount of overfitting, since the training IoU is much higher than the validation IoU. This suggests that the Pascal VOC 2012 dataset might actually be too small to train any of the proposed versions to their full extent. Hence, expanding the dataset used for training could improve the performance of both versions.

Since the version that uses partial masks has a much higher capacity than the one that does not (approximately 500M versus 150M learnable parameters, respectively), the size of the dataset necessary to avoid overfitting in this version will also be much greater. Hence, it could be possible that the simpler version outperforms the other primarily because the dataset used is too small for both, but even more impactfully smaller for the one using partial masks.

The primary way of testing this hypothesis is retraining and re-evaluating both of the versions using a larger dataset, for instance MS COCO. Unfortunately, due to limited time constraints, this investigation is left as a proposal for further work, described in section 5.3.

5.2 Error modes

In order to evaluate the impact of each subsystem to the quality of the final output, this section compares the average precision of the Masknet system for each class with the average precision that would be obtained if one of the subsystems (object detector, semantic segmenter, mask proposal) was able to produce perfect outputs.

This procedure is done for both versions of Masknet, the only difference being that the version that does not use partial masks is not affected by the semantic segmenter output, and hence this comparison is not required.

5.2.1 Masknet with partial masks

The Masknet version using partial masks was re-evaluated three times, each time using ground truth data to exchange one of its subsystems with the best possible output it could generate.

In the first re-evaluation, instead of using FCN-8s, the ground truth semantic segmentation is used to generate the semantic maps. These are then fed into the partial mask heuristic together with the predicted bounding boxes (the actual bounding boxes from Fast R-CNN, not ground truth data) to generate the partial masks. In this case, the system is retrained using this new, improved, input.

In the second one, both the semantic segmenter and the mask proposal subsystem are removed, and ground truth data for the semantic and object segmentations is used to generate perfect masks. This is achieved by choosing the ground truth object with the matching class and highest IoU for each predicted bounding box. This models the scenario where the system doesn't have perfect localization, but it can perfectly segment an object inside a bounding box.

Finally, the last re-evaluation is done by using ground truth data for the bounding boxes, instead of predicting them with Fast R-CNN. Both the semantic segmenter and the mask proposal subsystems are still in place. Figure 5.8 shows the absolute improvement in average precision for each one of the classes, for all re-evaluations.

As before, the conclusion is class-dependant. For instance, for the bicycle class the

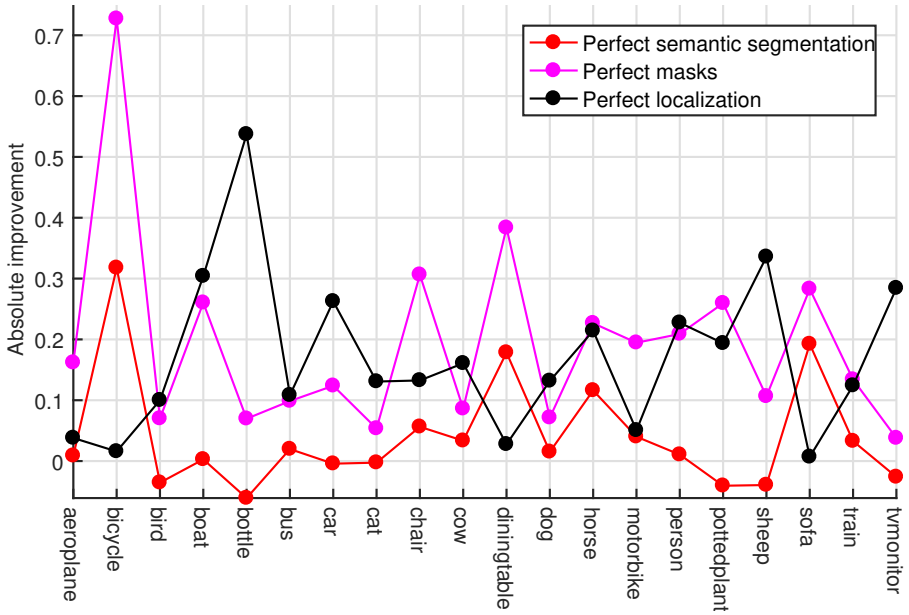


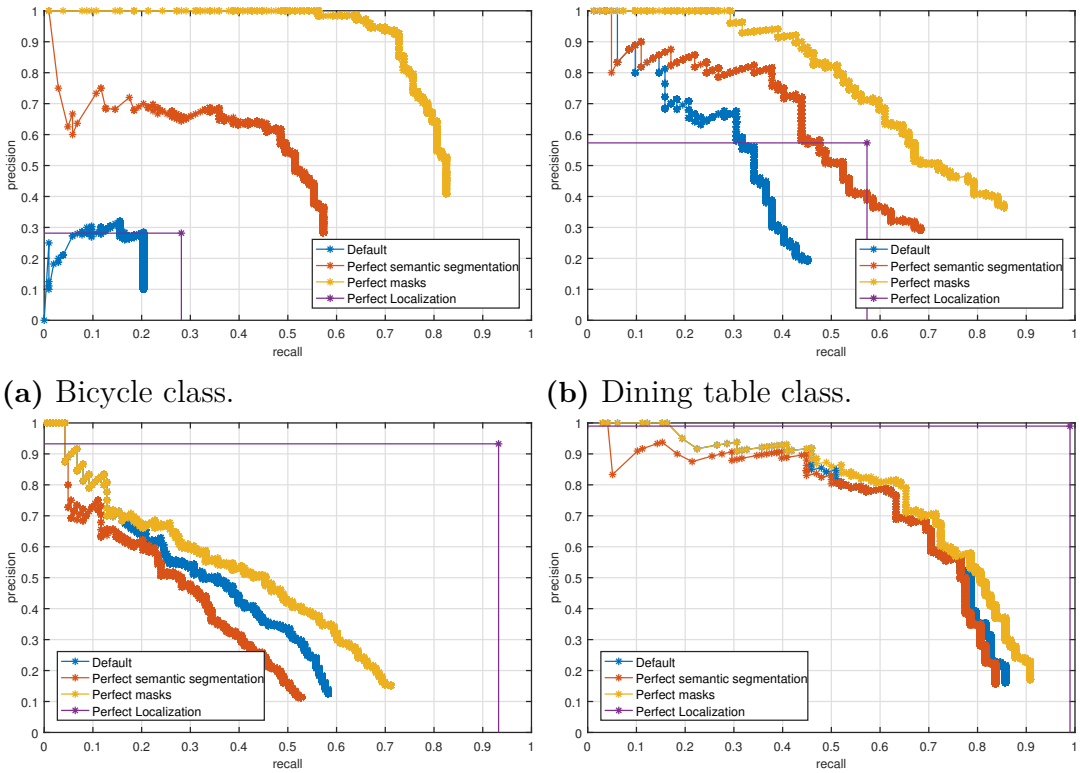
Figure 5.8: Absolute improvement in average precision (compared to original version) for each one of the classes, for all re-evaluations of the Masknet version using partial masks.

greatest improvement comes from using perfect masks, and secondly from using perfect semantic segmentations. This is very sensible, since the main problem with this class is segmenting it correctly. Figure 5.9a illustrates the precision-recall curve for this class. The curve for perfect localization has only one point because when perfect localization is used, the confidence levels for all bounding boxes are assigned 1, and thus the confidence levels for all instances are assigned 1, which generates only one precision-recall point.

The figure shows that indeed perfect masks and perfect semantic segmentations obtain a much better precision-recall curve, being able to almost perfectly segment bicycles until around 0.7 recall, when at that point precision falls abruptly. The figure also shows that improving localization has almost no impact on this class, since, as already explained, the problem is to correctly segment its members.

The same behaviour occurs with the dining table class. Its precision-recall curve is shown in figure 5.9b. For this and the sofa class, using perfect localization is apparently not helpful, possibly because perfect localizations for these objects include lots of extraneous instance parts, whereas incomplete localizations might cover only the object in question, thus facilitating the mask proposal's work.

With the bottle and T.V. monitor classes, which have their precision-recall curves illustrated in figures 5.9c and 5.9d, using perfect semantic segmentations is actually harmful to performance, resulting in an average precision that is lower than the one for the default system that does not use ground truth data. The reason for this is still unclear, but one possible explanation could be that the mask proposal subsystem learns to rely too much on the semantic segmentation, and when that fails (e.g. several bottles occluding each other) it is incapable of recovering from it. Using



(a) Bicycle class.

(b) Dining table class.

(c) Bottle class.

(d) T.V. monitor class.

Figure 5.9: Precision-recall curves for a subset of the Pascal VOC 2012 classes, for each re-evaluation performed.

perfect masks is beneficial but not substantially. Using perfect localization is the re-evaluation with the most impact, yielding almost perfect instance segmentations for these classes. This shows that if the bounding boxes are accurate enough, the system excels at segmenting these particular classes.

For a wider perspective of the performance across all classes, the MAP^r metric was also computed for each re-evaluation. The metric values for the default system, the perfect semantic segmentation, perfect localization, and perfect masks, are 0.5512, 0.5919, 0.7204 and 0.7443, respectively. This shows that there is still a lot of room for improvement of the mask proposal subsystem, while at the same time showing how substantial the improvement would be if a better object detector was used.

5.2.2 Masknet without partial masks

The Masknet version that does not use partial masks was also re-evaluated, each time using ground truth data to exchange one of its subsystems with the best possible output it could generate. However, since this version does not depend on the output of the semantic segmenter, it's only necessary to re-evaluate it twice.

In the first one, the entire mask proposal subsystem is removed, and ground truth data for the semantic and object segmentations is used to generate perfect masks.

This yields exactly the same re-evaluation as was done for the version that uses partial masks, since their only difference is in the actual mask proposal subsystem. The other re-evaluation is done by using ground truth data for the bounding boxes, instead of predicting them with Fast R-CNN. Figure 5.8 shows the absolute improvement in average precision for each one of the classes, for all re-evaluations.

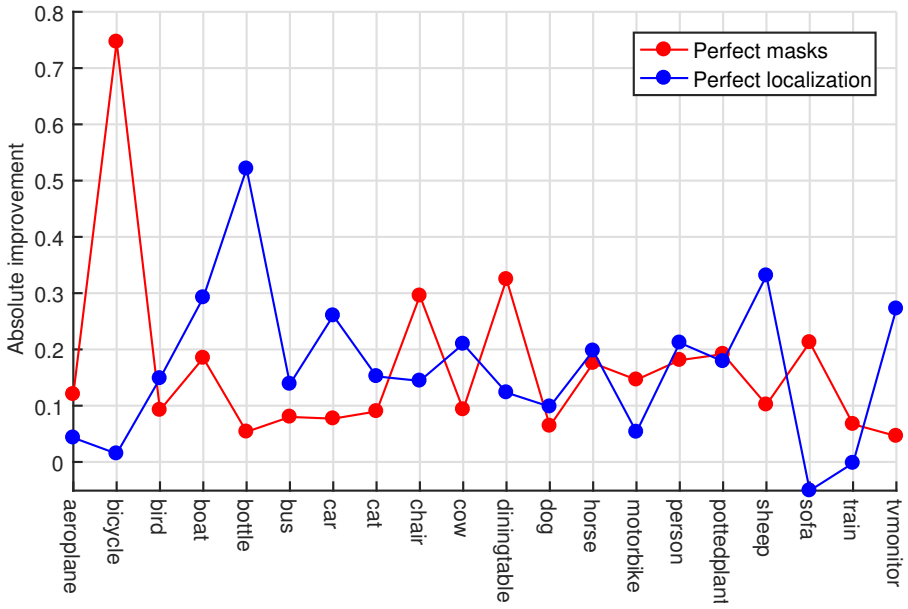


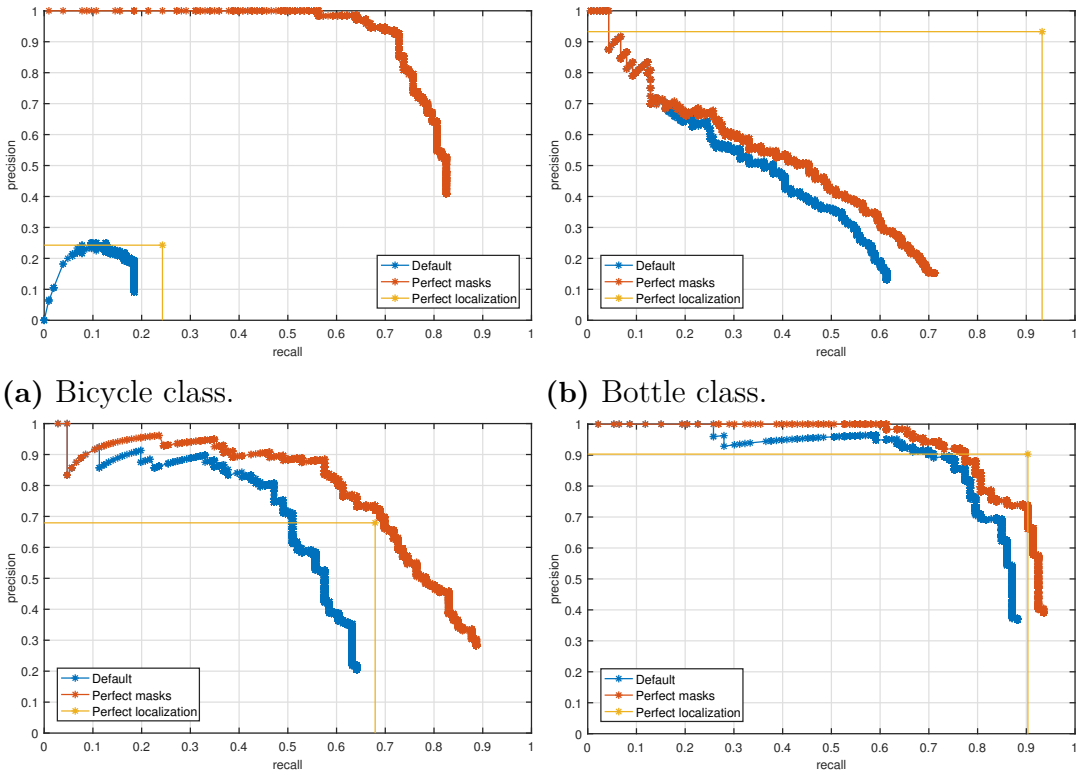
Figure 5.10: Absolute improvement in average precision (compared to original version) for each one of the classes, for all re-evaluations of the Masknet version that does not use partial masks.

Just like with the version that uses partial masks, the conclusion is class-dependant. In this version, the bicycle class is again the one with the highest improvement when using perfect masks, and the bottle class when using perfect localization, for the reasons cited before. Their precision-recall curves are illustrated in figures 5.11a and 5.11b.

This time, however, using perfect localization is actually harmful to performance for the sofa class. The precision-recall curve for this class is shown in figure 5.11c. As explained before, one possible explanation for this could be that perfect localizations for members of this class will contain lots of extraneous instance parts, which would in turn make the mask proposal's work harder, specially if no partial masks are provided.

Another interesting point is that perfect localizations almost didn't improve the average precision for the train class, which has its precision-recall curve shown in figure 5.11d. This possibly happens because the train instances present in Pascal VOC 2012 almost always occupy a large area in the scene, making them easy to detect and to segment. Hence, improving localization should not have a big impact in this class.

Just as before, the MAP^r metric is computed for each re-evaluation, in order to obtain a wider perspective of the performance across all classes. The metric values



(a) Bicycle class.

(b) Bottle class.

(c) Sofa class.

(d) Train class.

Figure 5.11: Precision-recall curves for a subset of the Pascal VOC 2012 classes, for each re-evaluation performed.

for the default system, the perfect masks and the perfect localization, are 0.5772, 0.7443, and 0.7440. These results show that, in the case that the object detection improves considerably, if the same dataset is used, both versions of Masknet will still perform similarly, but the versions without partial masks will still outperform the other. Furthermore, a comparison of the default performance with the one using perfect masks shows again that there is a lot of room for improvement on the mask proposal subsystem, just as with the version using partial masks.

5.3 Further work

This section is devoted to discussing some key improvements that could enhance the Masknet system further.

First of all, the single most important thing for improving the Masknet system's performance would be to re-train the object detector, the semantic segmenter and the mask proposal subsystems using a larger dataset, possibly the MS COCO 2016 dataset. As discussed in section 5.1, there is a great amount of overfitting happening to both versions of Masknet, and the primary reason is likely to be the small size of the Pascal VOC 2012 dataset. MS COCO has a much larger and more varied set of examples (around 300k images, compared to approximately 3k images for Pascal),

which could allow the systems to reach their maximum potential.

Secondly, although Fast R-CNN and FCN-8s achieve competitive results for images in the Pascal VOC dataset, they are not state of the art in their respective areas. It will be greatly advantageous to use better networks for these tasks, for instance Faster R-CNN [23] and PSPNet [31], if the hardware requirements are met.

Thirdly, it's likely that it would also be beneficial to try different architectures for the mask proposal subsystem besides the ones tested. Trying new variations of the DeepMask architecture, and also completely new designs, should prove to be very helpful in deciding if the main obstacle to a performance increase is just the size of the dataset being used. Furthermore, in such a scenario that several new architectures are being tested, it should be very advantageous to try a different, more efficient, procedure for optimizing the hyperparameters for each new network. For instance, one possibility could be to use random search for the hyperparameters, which was shown to be much better than grid search for most cases [32]. Additionally, this could effectively be used to explore different architectures (by interpreting design choices as new hyperparameters to be optimized).

Fourthly, as discussed in section 4.2, the confidence level assigned to each bounding box by the object detector is used to rank the instances when computing the MAP^r scores. Since each generated instance also depends on the output from the semantic segmenter, and the mask proposal subsystem, it seems logical that the confidence level for each instance should not depend only on the confidence level for its bounding box. It could be advantageous to combine information about the confidence of each subsystem about a given instance to better rank them when computing the metric. This way, for instance, the system might have a better ability to recover from a mislocalization, because the mask proposal subsystem could contribute to assigning an even lower confidence level to a badly predicted bounding box, making it contribute less to the final overall MAP^r score.

Finally, a very impactful problem for Masknet is badly predicted bounding boxes. A recent advance in binary mask prediction, named Boundary-aware instance segmentation [14], predicts binary masks from bounding boxes, but in such a way that the resulting mask is not restrained to the spatial extent of the bounding box. Implementing their ideas into a new Masknet architecture could make the system much more resilient to mislocalizations, and therefore increase its performance even further.

6

Conclusion

This thesis formulated, developed and evaluated two versions of an instance segmentation algorithm, Masknet. The first version used the output from an object detector CNN and a mask proposal subsystem to generate the instances, whereas the second one, to which three different possible architectures were devised, used also the output from a semantic segmenter to obtain initial guesses for the masks of each instance. All of the code developed for this thesis is made available at `https://github.com/JulianoLagana/masknet`.

The three different architectures devised for the Masknet version with partial masks were trained on data from Pascal VOC 2012 and evaluated by comparing the best IoU score on the validation set. The architecture that was best rated using this valuation was the one using merging strategy 3 (i.e. fully connecting the entire VGG-16 feature map and the heuristic partial mask to the neuron layer that computes the output vector). The other strategies also performed well but obtained weaker scores.

Then, by comparing the MAP^r metric score of the Masknet versions with and without partial masks, it was found that the best scoring one was the one without partial masks, with a obtained score of 57.7% (compared to 55.1% for the other version).

A visual assessment of the version with partial masks was conducted and it was found that for many examples the network did exactly what was expected, reconstructing the missing parts from the partial masks (or removing incorrect ones). However, it was also found that the system is unable to recover from strong mislocalizations (i.e. very badly predicted bounding boxes), and for some examples, even in relatively favourable conditions, the system failed to generate a reasonable mask. As discussed in section 5.1, this and the resulting plots for the loss and IoU in the training/validation set led me to hypothesize that the reason for this is overfitting, but the verification of this hypothesis is left as further work.

Finally, the impact of each subsystem to the quality of the final output was assessed for each version of Masknet. This showed that there is still room for improvement in the mask proposal subsystem, and enhancing its performance can lead to better scores in particular classes, like the bicycle class for instance. Another important finding is that both versions can greatly improve their MAP scores by using a better object detector, like the one suggested in section 5.3. Lastly, for the version using partial masks, improving the semantic segmenter might not yield such a large benefit, and should perhaps be seen as a secondary step towards improvement.

Bibliography

[1] Dan Ionescu, Bogdan Ionescu, Cristian Gadea, and Shahidul Islam. An intelligent gesture interface for controlling tv sets and set-top boxes. *2011 6th IEEE International Symposium on Applied Computational Intelligence and Informatics (SACI)*, 2011.

[2] Tolga Soyata, Rajani Muraleedharan, Colin Funai, Minseok Kwon, and Wendi Heinzelman. Cloud-vision: Real-time face recognition using a mobile-cloudlet-cloud acceleration architecture. *2012 IEEE Symposium on Computers and Communications (ISCC)*, 2012.

[3] Alessandra Maria Sabelli, Takayuki Kanda, and Norihiro Hagita. A conversational robot in an elderly care center: An ethnographic study. In *Proceedings of the 6th International Conference on Human-robot Interaction*, HRI '11, pages 37–44, New York, NY, USA, 2011. ACM.

[4] M.a. Turk, D.g. Morgenthaler, K.d. Gremban, and M. Marra. Vits - a vision system for autonomous land vehicle navigation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(3):342–361, 1988.

[5] Bharath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. Simultaneous detection and segmentation. *Computer Vision – ECCV 2014 Lecture Notes in Computer Science*, page 297–312, 2014.

[6] Bernardino Romera-Paredes and Philip Hilaire Sean Torr. Recurrent instance segmentation. *Computer Vision – ECCV 2016 Lecture Notes in Computer Science*, page 312–329, 2016.

[7] Anurag Arnab and Philip Torr. Bottom-up instance segmentation using deep higher-order crfs. *Procedings of the British Machine Vision Conference 2016*, 2016.

[8] Bharath Hariharan, Pablo Arbelaez, Ross Girshick, and Jitendra Malik. Object instance segmentation and fine-grained localization using hypercolumns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(4):627–639, Jan 2017.

[9] Jifeng Dai, Kaiming He, and Jian Sun. Instance-aware semantic segmentation via multi-task network cascades. *CoRR*, abs/1512.04412, 2015.

[10] Yi Li, Haozhi Qi, Jifeng Dai, Xiangyang Ji, and Yichen Wei. Fully convolutional instance-aware semantic segmentation. *CoRR*, abs/1611.07709, 2016.

[11] Alireza Fathi, Zbigniew Wojna, Vivek Rathod, Peng Wang, Hyun Oh Song, Sergio Guadarrama, and Kevin P. Murphy. Semantic instance segmentation via deep metric learning. *CoRR*, abs/1703.10277, 2017.

[12] Guanbin Li, Yuan Xie, Liang Lin, and Yizhou Yu. Instance-level salient object segmentation. *CoRR*, abs/1704.03604, 2017.

[13] Anurag Arnab and Philip H. S. Torr. Pixelwise instance segmentation with a dynamically instantiated network. *CoRR*, abs/1704.02386, 2017.

[14] Zeeshan Hayder, Xuming He, and Mathieu Salzmann. Shape-aware instance segmentation. *CoRR*, abs/1612.03129, 2016.

[15] Mengye Ren and Richard S. Zemel. End-to-end instance segmentation and counting with recurrent attention. *CoRR*, abs/1605.09410, 2016.

[16] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. *CoRR*, abs/1703.06870, 2017.

[17] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149, Jan 2017.

[18] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111(1):98–136, January 2015.

[19] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.

[20] Michael A. Nielsen. *Neural Networks and Deep Learning*. Determination Press, 2015.

[21] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[22] Chris J.C. Burges. A tutorial on support vector machines for pattern recognition. volume 2, pages 121–167, January 1998.

[23] Ross B. Girshick. Fast R-CNN. *CoRR*, abs/1504.08083, 2015.

[24] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *CoRR*, abs/1411.4038, 2014.

[25] Pedro O Pinheiro, Ronan Collobert, and Piotr Dollar. Learning to segment object candidates. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 1990–1998. Curran Associates, Inc., 2015.

[26] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages

249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR.

[27] Christopher D. Manning, Prabhakar Raghavan, and Schütze Hinrich. *Introduction to information retrieval*. Cambridge University Press, 2009.

[28] João Carreira, Rui Caseiro, Jorge Batista, and Cristian Sminchisescu. Semantic segmentation with second-order pooling. *Computer Vision – ECCV 2012 Lecture Notes in Computer Science*, page 430–443, 2012.

[29] Yi-Ting Chen, Xiaokai Liu, and Ming-Hsuan Yang. Multi-instance object segmentation with occlusion handling. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[30] Jifeng Dai, Kaiming He, and Jian Sun. Convolutional feature masking for joint object and stuff segmentation. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[31] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. *CoRR*, abs/1612.01105, 2016.

[32] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *J. Mach. Learn. Res.*, 13:281–305, February 2012.