University of Warsaw
Faculty of Mathematics, Informatics and Mechanics

Mateusz Biegański
Student no. 385162

Kamil Bladoszewski
Student no. 392558

Piotr Gierda
Student no. 385507

Paweł Opiela
Student no. 385926

Multi-Task Learning in Computer Vision using PASCAL in Detail Dataset

Bachelor's thesis
in COMPUTER SCIENCE

Supervisor:
mgr Grzegorz Grudziński

June 2019

Supervisor's statement

Hereby I confirm that the presented thesis was prepared under my supervision and that it fulfils the requirements for the degree of Bachelor of Computer Science.

Date Supervisor's signature

Authors' statements

Hereby I declare that the presented thesis was prepared by me and none of its contents was obtained by means that are against the law.

The thesis has never before been a subject of any procedure of obtaining an academic degree.

Moreover, I declare that the present version of the thesis is identical to the attached electronic version.

Date Authors' signatures

Abstract

In this work we present a modified version of Facebook Research's implementation of Mask R-CNN with added PASCAL in Detail dataset support. We deem the extra feature beneficial, as it enables the use of annotations for 9 different computer vision tasks. Two main technical challenges that emerged during the making of this architecture are addressed, namely conflicting data formats and multi-task learning. The changed system with an added segmentation head now handles object detection, human keypoint estimation, instance segmentation and semantic segmentation.

Keywords

multi-task learning, PASCAL in Detail dataset, Mask R-CNN, object detection, semantic segmentation, instance segmentation, human keypoint estimation, convolutional neural networks

Thesis domain (Socrates-Erasmus subject area codes)

11.3 Informatics, Computer Science

Subject classification

Computing methodologies
- Machine Learning
  - Learning paradigms
    - Multi-task learnig
- Aritificial Intelligence
  - Computer vision
    - Computer vision problems
      - Image segmentation

Tytuł pracy w języku polskim

Wielozadaniowe uczenie w wizji komputerowej przy użyciu bazy danych "PASCAL in Detail"

Contents

Introduction 5

1. PASCAL in Detail 7
1.1. Short summary of tasks 7
1.2. PASCAL in Detail dataset 10
1.3. Segmentation classes 14

2. Initial Architecture 15
2.1. Overview of Mask R-CNN 15
2.2. Benefits of modification 16

3. Multi-Task Learning 17
3.1. Importance 17
3.2. Source code implementation 17
3.3. PASCAL in Detail support 18
3.4. Filtering 18
3.5. Results 21

4. Semantic Segmentation 23
4.1. Task description 23
4.2. Why semantic segmentation? 23
4.3. Applications 23
4.4. Results 23
4.5. Configuration API 23
4.6. Design decisions 24
4.6.1. Dataset preparation 24
4.6.2. Predictor 24
4.6.3. Loss evaluator 24
4.7. Technical problems 25
4.8. Prospects for continuation 25

5. Development & Operations 27
5.1. Deployment 27
5.2. Unused machines 28
5.3. Training 28

6. Training Results . . . 29
6.1. Results . . . 29
6.2. Mean Average Precision . . . 29
6.3. Interpretation . . . 29

7. Conclusion . . . 31
7.1. What was the goal? . . . 31
7.2. What did we accomplish? . . . 31
7.2.1. Future possibilities of development . . . 31

8. Work Division . . . 33

9. Content of the CD . . . 35
9.1. Repository structure . . . 35
9.2. Installation . . . 35

A. Annotation Example . . . 37

B. Result Example . . . 39

Bibliography . . . 41

Introduction

The emergence of the PASCAL in Detail [1] dataset led to new possibilities in creating deep learning algorithms that solve many different computer vision problems at once. Supplying annotations for not only basic tasks like detection or classification, but also more profound ones such as boundary or occlusion detection, a new rich multi-task architecture becomes possible. A full description of the data and the task annotations which it provides is in chapter 1. To fully utilize this new dataset we wanted to use some existing architecture and change the dataset on which it works. Our client suggested Facebook Research's implementation of Mask R-CNN [10], which uses the COCO Dataset [6]. Coming from a well know research group, it was deemed credible. Specifics of the exact version of Mask R-CNN, which we used, are in chapter 2.

As mentioned in the abstract, the two main technical challenges that emerged were conflicting data formats and multi-task learning. Mainly there are differences in annotations for instance segmentation. The COCO dataset uses polygons and PASCAL in Detail uses RLE format. The code from Facebook required one general format of annotations which was problematic due to the task specific annotations in PASCAL in Detail. This was a challenge during implementing multi-task learning, which was solved by appropriate annotation filtering. How we overcame the described difficulties is shown in chapter 3. Also initial stages of a new semantic segmentation head were implemented. The details of this new component of Mask R-CNN are in chapter 4.

The main purpose for this project was to lay the groundwork of a future well developed multi-task architecture. We wanted to have as many tasks implemented as possible. This would enable many future applications and possible opportunities of development of our product. A summary of our work is presented in chapter 7.

Chapter 1

PASCAL in Detail

The PASCAL In Detail dataset was initially released in 2017 during Conference on Computer Vision and Pattern Recognition [1], which took place in Honolulu, Hawaii, USA. It is based on the PASCAL VOC 2010 Challenge dataset. Later it was enriched with more tasks and annotations, which made the dataset more versatile and suited for multi-task learning.

The PASCAL VOC dataset was released for the first time in 2005 and since then was used in 8 VOC Challenges, which were focused mainly on classification, detection and semantic segmentation. The final PASCAL VOC Challenge took place in 2012 [2]. The word *PASCAL* stands for pattern analysis, statistical modelling and computational learning, while *VOC* stands for visual object classes [3].

1.1. Short summary of tasks

Before we introduce the PASCAL In Detail dataset we would like to provide a short summary of the tasks for which the dataset provides annotations.

1. *Semantic Segmentation* is the task of labelling each pixel of an image with a certain class. The task is also known as *Image Segmentation*

2. *Instance Segmentation* is the task of identifying individual object instances. The difference between semantic segmentation is that different instances of the same class have different labels.

3. *Part Segmentation* is the task of taking an object instance and identifying which pixel of this instance belongs to which part of the object e.g., finding the door of a car.

4. *Boundary Detection* is the task of finding boundaries between semantic classes.

5. *Occlusion Recognition* is the task of determining which object instance, between two neighboring objects, is in front of the other.

6. *Human Keypoint Estimation* is the task of identifying important points of an object e.g., the head of a human. The name can be used interchangeably with *Person Keypoint Detection* and the varations of these two names.

7. *Image Classification* is the task of presenting a list of classes present in an image.

8. *Object Detection* is similar to image classification, except it provides also a boundary box for each object instance of a certain class that is present.

9. *Objectness* is the task of determining if there is any non-background object present in an image.

(a) Input

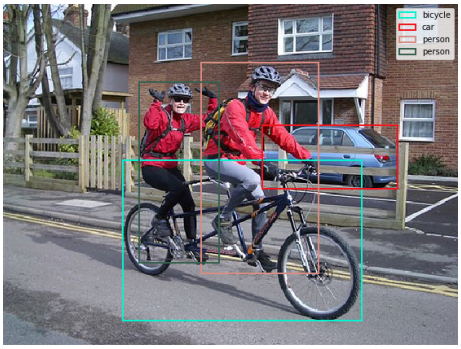
(b) Semantic Segmentation

(c) Instance Segmentation

(d) Part Segmentation

(e) Boundary Detection

(f) Occlusion Recognition



(g) Object Detection

(h) Keypoint Estimation

Figure 1.1: Visualization of task annotations taken from the PASCAL in Detail dataset

1.2. PASCAL in Detail dataset

Our project aims to tackle the multi-task learning problem. Because of it, we favoured to use the PASCAL in Detail dataset over the COCO dataset, which is built into the initial architecture described in chapter 2. Unfortunately, the COCO dataset does not provide annotations for as many tasks as we would like. The Detail dataset is more diverse and contains much more tasks. Additionally, most of the images have all the types of annotations, so we can use basic synchronous stochastic gradient descent (**SGD**) and are not forced to use more complicated asynchronous SGD as in [4].

Synchronous SGD is commonly used in the process of training neural networks. One training epoch (iteration over a dataset) is divided into mini-batches of training samples (in case of computer vision: images and their annotations). The network performs backpropagation after each mini-batch [5]. This may lead to problems if some tasks do not have annotations for the current image. Asynchronous SGD performs backpropagation after a certain number of samples for each of the tasks has been seen by the network. The concept is further described in [4].

Even though we do not use all of the annotations, further research and implementation of the rest of the tasks is possible thanks to our work on the network.

Task	COCO [6]	Detail [1]
Classification	NO	YES
Detection	YES	YES
Semantic Segm.	YES	YES
Instance Segm.	YES	YES
Objectness	NO	YES
Part Segm.	NO	YES
Boundary	NO	YES
Occlusion	NO	YES
Keypoint	YES	YES
Captioning	YES	NO

Table 1.1: Comparison of annotations present in the datasets

As a disadvantage of this approach, we can mention a smaller number of pictures and annotations. The COCO dataset has over 200k labeled images and 1.5M annotations [6] in comparison to 10.1k photos and 114k annotations in the Detail dataset. Even though it might be a problem in end-to-end training, we are only fine-tuning the task specific heads. This approach requires much less data and relies on the convolutional backbone trained on ImageNet.

ImageNet is an image database commonly used in the field of computer vision. It is often utilized in the training of the convolutional neural networks, which are then fine-tuned using more specific data [7].

As already mentioned, the Detail dataset provides annotations for 9 types of possible tasks for the neural network, which is the most important reason to use it in our project. As mentioned in the beginning of chapter 1, the annotations are based on the PASCAL VOC dataset [1]. The annotations are as follows:

1. *Instance Segmentation* uses 20 classes from PASCAL VOC dataset. Creators: Xianjie Chen and Xiaochen Lian.

2. *Semantic Segmentation* consists of 59 classes, out of which 20 come from PASCAL VOC dataset. Creators: Xiaochen Lian and Seong-Whan Lee.

3. *Part Segmentation* uses 16 out of 20 original PASCAL VOC classes. Creators: Xianjie Chen and Peng Wang.

4. *Boundary Detection* makes use of 59 classes used in Semantic segmentation task. Creator: Vittal Premachandran.

5. *Occlusion Recognition* Creator: Peng Wang

6. *Keypoint Estimation* Creator: Fangting Xia

7. *Image Classification* consists of 59 classes used in semantic segmentation and is generated from these annotations. The task is designed to be a multi-class recognition problem, which means that each image is associated with 59 booleans – whether the class is present in the picture or not.

8. *Object Detection* consists of 20 categories from PASCAL VOC dataset and is generated from instance segmentation annotations.

9. *Objectness*, just as object detection, uses annotations and 20 classes from semantic segmentation.

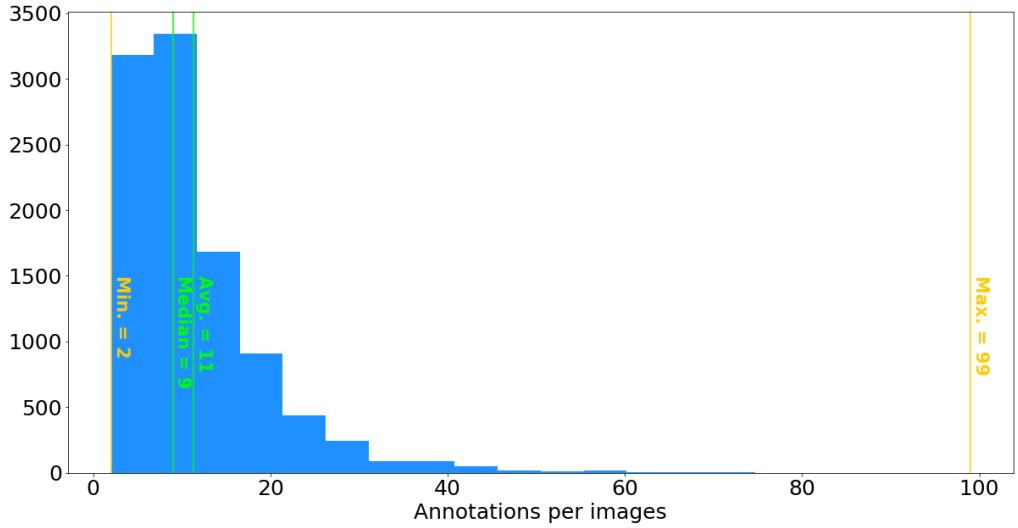


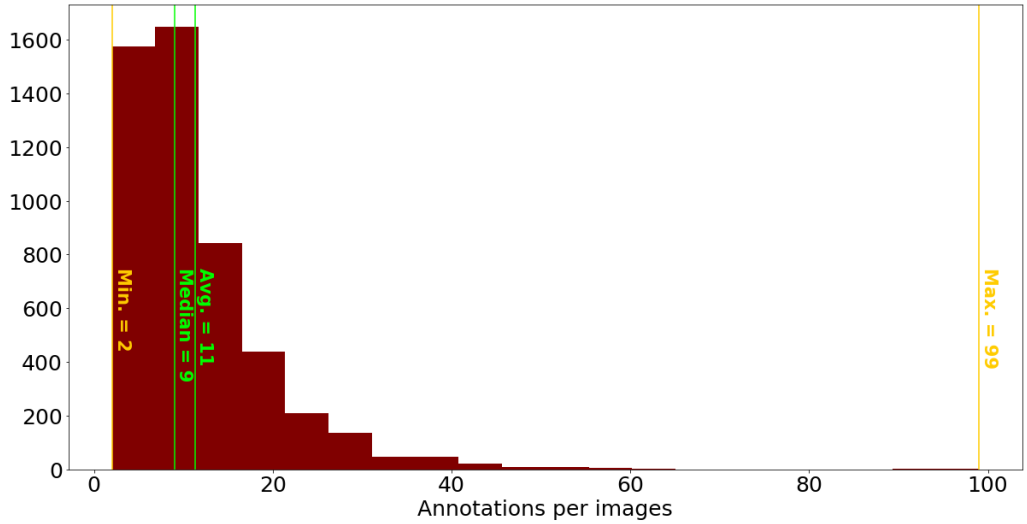
Figure 1.2: Histogram depicting the number of annotations per photo in the Pascal in Detail dataset

The Detail dataset comes with a training and validation subset division. It is beneficial to our work as we can easily train and then test our model on different images. This way, we can be sure, that the testing results are not biased and our predictions are bona fide. Furthermore, the already split dataset saved us time by not having to do it ourselves.

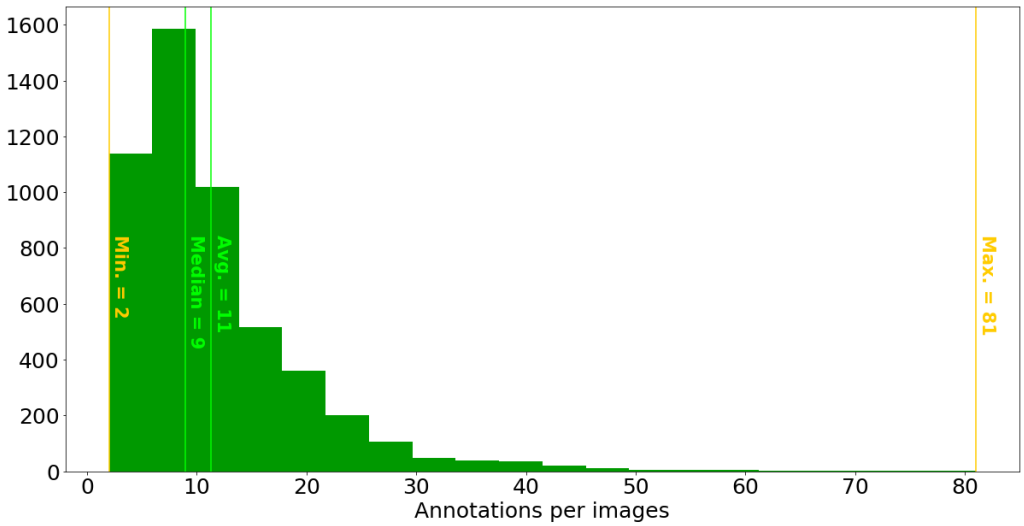
The whole dataset consists of 10100 annotated images and 114425 annotations. The train subset contains 4996 images and 56670 annotations. The validation dataset contains 5104 images and 57755 annotations.

Annotations are rather evenly split between the images, however, there are some anomalies of images, which are more annotated than the others. Most of the images have 4 annotations

and an image has over 11 annotations on average, but half of them have less than 9 annotations. Based on this information, we can assume that there are a few outliers and the data is widely spread. This can lead to worse generalization, which is a disadvantage, but on the other hand, enough outliers can allow our model to learn uncommon data.



(a) Training subset



(b) Validation subset

Figure 1.3: Histograms depicting the number of annotations per photo in subsets

The annotations are divided into 4 categories: *segmentation*, *boundary*, *occlusion*, and *keypoints*. The vast majority of the annotations correspond to the segmentation tasks (86935 annotations). Occlusion and boundary annotations are far behind (10094 and 10100 annotations respectively), as an image has one such annotation at most. Since most of the images do not depict any people, the smallest category is keypoints – it contains only 7296 annotations.

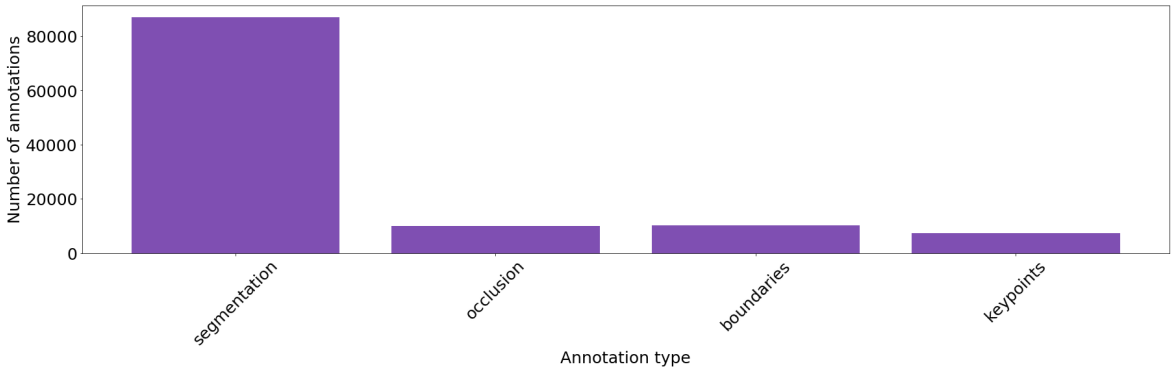
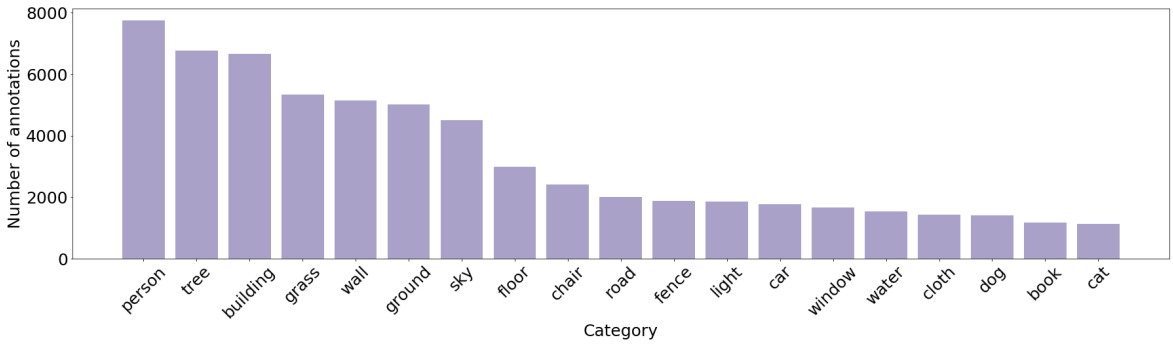
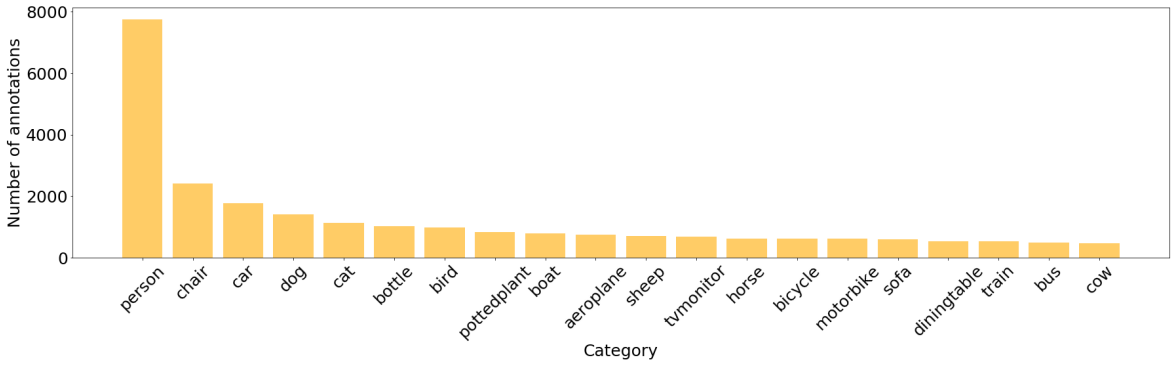


Figure 1.4: Number of each type of annotations



(a) Semantic Segmentation classes



(b) Instance Segmentation classes

Figure 1.5: Number of instances of top 20 segmentation classes

1.3. Segmentation classes

As mentioned before, the Detail dataset contains 59 segmentation classes, of which 39 are used only for semantic segmentation, and 20 are used for both semantic and instance segmentation [1]. It is worth noting that most of the segmentation annotations are semantic-only. It is not only the result of the bigger number of semantic-only classes, but also the difference in the number of instances of objects from semantic-only classes.

The class for instance segmentation can be any object, which can have multiple instances on an image, e.g., cat or book. However, looking at the the figure 1.5, you may note that *book* category appears only in the semantic segmentation part. This can lead to a conclusion that *book* is a semantic-only class, even though it can be divided into instances.

Transforming semantic segmentation annotations into instance segmentation annotations requires human labelling. Because of this, the creators of the dataset have to balance between the efficiency of the process and the completeness of the dataset. It appears that in the case of this dataset, the creators decided to omit some less important classes.

What is worth mentioning, classes which cannot be divided into instances also will not be present in the annotations for instance segmentation. This includes background, e.g., sky, ground, and grass.

The distribution of instances over classes is also worth analysis. As shown in figure 1.5, the category *person* appears the most often and is used for both semantic and instance segmentation. This may imply that a network trained on the dataset will slightly better recognize people, especially in the instance segmentation task.

The figure 1.5 also depicts the gap between the number of instances of semantic-only classes and the ones used also in the instance segmentation task. Only the top 5 instance segmentation classes are among the 20 most frequent semantic segmentation classes.

Chapter 2

Initial Architecture

Knowing the potential of the PASCAL in Detail dataset we wanted to find an available architecture, which could exploit the large amount of possible annotation types. Facebook Research's Mask R-CNN [10] was proposed by our client as a great candidate and we agreed. The reason for this was that it already had state-of-the-art results in the object detection, instance segmentation and human keypoint detection tasks. Initially, this architecture was implemented in the Caffe2 deep learning framework. However, there was a Pytorch version in its earlier stages, which was preferred by our client and us. This was because this framework was deemed more appropriate for research applications. In this section, we would like to describe the architecture, which was used to apply the PASCAL in Detail dataset.

2.1. Overview of Mask R-CNN

Mask R-CNN is a deep learning architecture, meaning it uses convolutional neural nets to create predictions. *Mask* means that it creates segmentation masks, which are annotations containing information to which class each pixel belongs. They are used in e.g., semantic segmentation or instance segmentation. Also this element separates the neural net from its predecessor Faster R-CNN, which did not perform any segmentation tasks. *R-CNN* is short for "Regions with convolutional neural net features", meaning it takes the output of the CNN and divides it into parts corresponding to regions of the image where it supposes there is an object. A more in-depth view of the components of this architecture is given in figure 2.1.

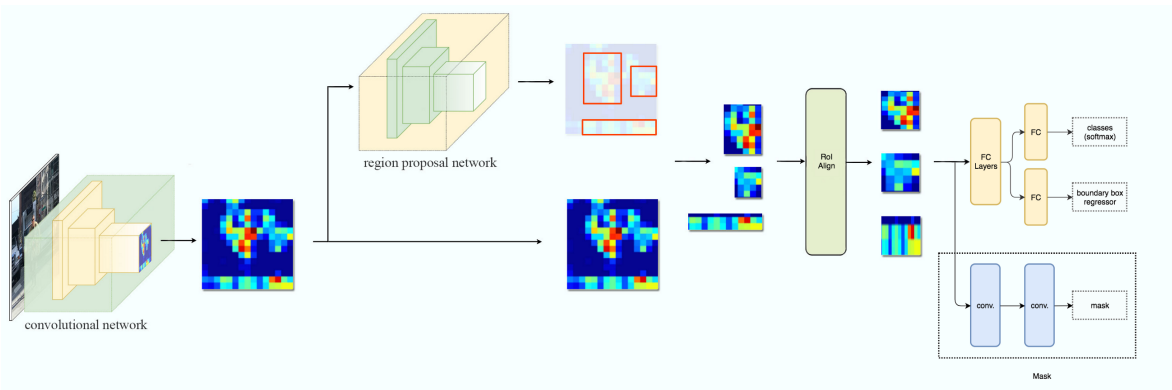


Figure 2.1: Mask R-CNN Architecture [12]

The first step in image processing by Mask R-CNN is extracting feature maps, using any known implementation of a convolutional neural net. In our case, it was Resnet-50 with Feature Pyramid Network (FPN). What is really important is the usage of the FPN module, which significantly improves the quality of the whole network. As mentioned above, the convolutional neural net produces a feature map based on the photo. With the FPN, it is a tuple of feature maps with different resolutions. Later, depending on the task, this is fed into the region proposal network (RPN), or, like in semantic segmentation, it goes straight to the task head. In the RPN, regions likely containing an object are produced. They are also called regions of interest (RoIs). Next, they are warped into the same size by the ROI Align module. At this stage, the predictions are ready to be analyzed by the task-specific heads. In figure 2.1 the dotted lines present the instance segmentation head and above is the object detection head. For image clarity, the human keypoint detection head is not present. They are additional convolutional nets that are only concerned with their designated task. This means that during training, they learn only task-specific characteristics. During multi-task learning and inference, time would be saved due to the common backbone (Resnet-50). The modular design also enables the architecture to be only trained for tasks that are needed. These traits made Mask R-CNN a great fit for our purposes.

2.2. Benefits of modification

A question that may come to mind is: What can modifying such a great architecture achieve? The answer comes down to the dataset on which it runs and future possibilities. The implementation of Mask R-CNN which we used had COCO dataset support. As mentioned in chapter 1, this dataset provides annotations for much fewer types of computer vision problems. Also due to the modularity of the design, adding task-specific heads for each of the remaining 6 tasks is very straightforward.

This is why substituting it with PASCAL in Detail enables future developments of our solution that may even solve all available tasks. Our implementation can be seen as the foundation of such projects.

Chapter 3

Multi-Task Learning

Multi-task learning is a subfield of machine learning training that allows for solving multiple tasks at once, using a single model and taking advantage of the tasks' similarities to perform it efficiently. This approach enables getting predictions for multiple different tasks with a single inference.

3.1. Importance

This approach was proven faster for tasks which were somewhat related than learning each of the tasks on a separate net and then merging the results. Another powerful quality of multi-task learning is its ability to generalize between the tackled tasks. It is very useful when for example each of the tasks has too little annotated data to train a deep net on it, but collectively they are enough to train a common bigger part of the model and also smaller task-specific segments.

Multi-task learning revolutionized many fields e.g., autonomous vehicles and robotics, because it allows their decision-making systems to efficiently analyze input from all outer world sensors with a single net inference, saving both memory usage and computation time.

Significant improvements in performance regarding international web-search ranking datasets enabled the inception of projects such as RoboEarth, which aspire to create an open source internet database accessible worldwide not only for humans, but also various AI agents.

The aforementioned ability of generalization between tasks proved to be strongly beneficial especially in fields where gathering lots of labeled data can be particularly difficult, e.g., medical image analysis [11].

It is no surprise that many multi-task learning conferences emerged in recent years, including the one that inspired us to tackle this challenge - Pascal In Detail.

3.2. Source code implementation

Facebook Research's Pytorch implementation of Mask R-CNN [10] (architecture described in section 2, tasks in section 1.1) allowed for learning in two 'modes' using the COCO dataset annotations.

Using only annotations for the human class - learning all 3 tasks at once, i.e human keypoint detection, instance segmentation, object detection.

Using annotations for all 20 non-semantic-only classes available in the COCO dataset - learning 2 of the tasks, i.e., instance segmentation, object detection.

The second mode's limitation comes from the Mask R-CNN characteristic that all annotations have to be of identical format, i.e., have exactly the same fields of data.

Another downside was the lack of support for segmentation annotations represented in RLE format. Precisely, segmentation annotations come with the iscrowd boolean flag with 2 possible values, i.e.:

1 - this annotation represents a lot of objects and RLE is used

0 - this annotation represents a single object and polygons are used

Both RLE and polygons are described in section 3.4 within the instance segmentation description.

The struggle was that the COCO dataset made the aforementioned distinction, while the Detail dataset had all segmentation annotations represented as compressed, encoded RLE. Even though the COCO dataset provided data with either RLE or polygons, the source Mask R-CNN implementation filtered out any annotations with iscrowd flag value not equal 0.

What it meant for us is that the only acceptable segmentation annotation format was polygons which effectively disallowed any segmentation annotations from the Detail dataset.

3.3. PASCAL in Detail support

While implementing PASCAL in Detail support the major difficulties we encountered were the differences between COCO's and Detail's annotations. Mask R-CNN expects annotations for each task in every observation and, while COCO delivered annotations exactly in that manner, Detail had each task's annotations in a separate table.

We could not replicate COCO's behavior because for a given picture's annotations, which were in different tables within the Detail dataset, they did not correspond to each other. That is why we had to create an annotation format unifying all our observations and emulating the COCO-like Mask R-CNN annotation format.

On account of that, we were forced to use placeholders where our data for a given object was not sufficient. Allowing the net's heads to train on dummy data would lead to them learning with an unwanted bias. To prevent that from happening we implemented many data filters upon which we omitted the aforementioned placeholders as well as invalid or insufficient annotations already present in the Detail dataset.

3.4. Filtering

We would evaluate each annotation for a given picture omitting invalid ones and if none of the annotations were valid we would discard that picture with its observations altogether.

Annotation and photo filtering were highly task-specific, so we would carry it out for each head separately. These are the details for each head's filtering:

1. *object detection*

   Along with the 20 classic PASCAL VOC classes, the dataset also has 39 additional ones intended only for semantic segmentation, so we had to exclude annotations belonging to any of those additional classes.

   Placeholder: a list containing an empty list simulating polygon data format

2. *instance segmentation*

   Two of the more popular segmentation annotation data formats and the ones used in either the COCO or Detail dataset are:

polygons - representing a segmentation mask as a list of lists of coordinates creating a closed polygonal chain. The chain is a given object part's boundary and its inside is this object part's area. Each list represents a part of the object and all lists account for the whole object.

RLE (Run-Length Encoding) - representation of a mask is a 2D matrix of ones for the pixels/parts of an image where the given class is present and zeros otherwise. But that's not all. In our case, when RLE is mentioned with regards to the COCO or Detail dataset, it actually means that the aforementioned representation is encoded firstly with RLE then with LEB128 (Little Endian Base 128) resulting in an annotation consisting of a mask's size and a string of double encoded bytes.

We wrote a script converting segmentation annotations from RLE to polygons to make RLE represented annotations from the Detail dataset compatible with the polygon-only Mask R-CNN implementation.

On top of the class-based filtering mentioned in 3.4.1, we had to make sure polygons representing a given object instance's segmentation were containing at least 3 points i.e., 6 floats so that the area was positive. Six is the minimal acceptable value, because part of an object represented as an empty list (0 floats), a point (2 floats) or a line (4 floats) would have a nonexistent area which makes an annotation like that counterproductive. The polygon is a list of coordinates so the number of floats also has to be even.

Placeholder: a list containing an empty list simulating polygon data format

3. *human keypoint detection*

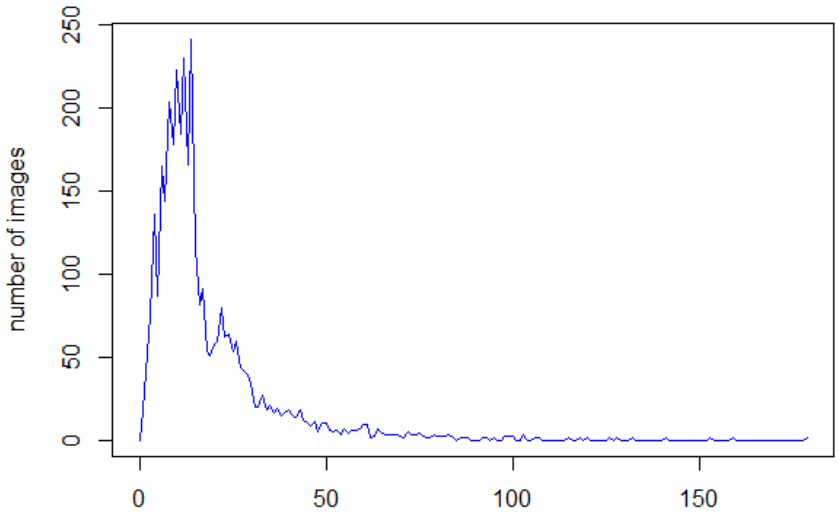
   We could only accept annotations of class 'person'.

   There was a difference in skeletons, amount and specifics of keypoints provided between the Detail dataset and the Mask R-CNN expected keypoint format.

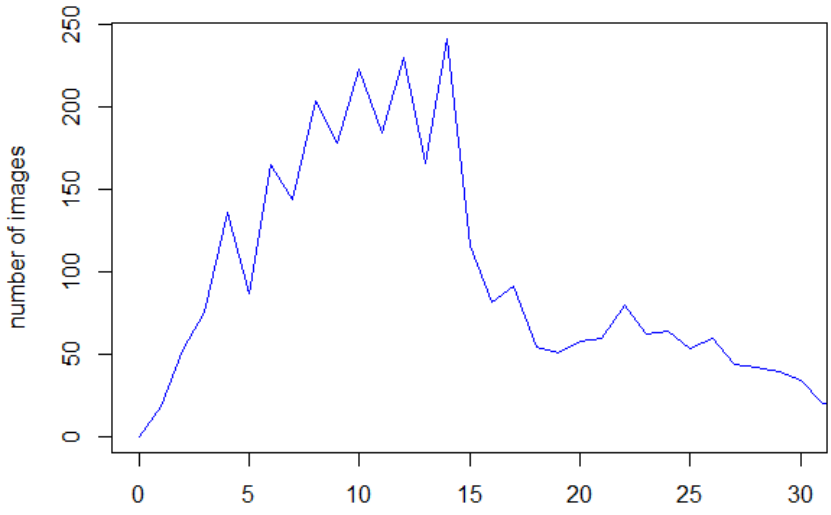
   Some of the keypoints types in the original Mask R-CNN implementation are not present in the Detail dataset i.e.: 'nose', 'left eye', 'right eye', 'left ear', 'right ear'. Instead, this part of a human body is described using 'head' and 'neck' types. This change required adjustments in skeleton size, expected number of keypoints per person, keypoint connections and their flip maps.

   Another part of this filtering was making sure that a minimal number of keypoints are present and visible in a picture, unless it would be filtered out. In the original implementation, the threshold for the lowest allowable keypoints number in a given image was 10. Due to the fact that a significant portion of our annotations had fewer keypoints than required, we decided to lower the limit to 5, which based on the Detail dataset's keypoint statistical analysis, looks like a reasonable value to choose as seen in figure 3.1.

   Placeholder: a list of 42 zeros representing 14 keypoints, each of them defined as a trio of numbers. The first two are given keypoint's coordinates (0,0) and the third one is a boolean keypoint validity flag set as 0. What this flag being equal to zero means is that the keypoint it corresponds to is phony and, in particular, during training should be considered as such and ignored.



(a) Number of keypoints



(b) The graph above zoomed in

Figure 3.1: histogram of keypoints number per image

3.5. Results

As a result of utilizing our script to convert annotations between the Detail dataset's and Mask R-CNN compatible formats combined with extensive annotation filtering, we managed to enable training all 3 tasks with annotations of all 59 available classes, which is a significant improvement compared to the available modes of training of the initial architecture. The RLE to polygon conversion was implemented as part of the aforementioned script.

Furthermore, we laid the ground for introducing new net heads requiring arbitrarily many task-specific annotations. Notably, the tasks can be the ones for which the Detail dataset provides annotations.

Chapter 4

Semantic Segmentation

4.1. Task description

As a reminder, semantic segmentation is the task of labelling each pixel of an image with a certain class, i.a. tree or car. It allows us to recognize the whole image environment, as there are no pixels without annotations. Most semantic segmentation implementations cannot distinguish between objects (lack of instance-awareness).

4.2. Why semantic segmentation?

Among all tasks to-be-implemented we chose semantic segmentation, because of it's multiple applications, much wider than any other task.

4.3. Applications

Taking into account the high potential of semantic segmentation (total comprehension of the environment), it may have a variety of real-life applications, some of which are presented below.

- Autonomous cars (environment understanding)
- Medical imaging
- Robot perception systems

4.4. Results

The Mask R-CNN implementation we used did not contain a semantic segmentation module. The code associated with it was produced by us. We have built a large part of it, starting from extracting image masks from the dataset, processing them, and outputting a mask the size of the picture, with each pixel labeled with a class number. Our implementation lacks an evaluation module, which would interpret the accuracy of the obtained results.

4.5. Configuration API

We added new fields to the network configuration file, all associated with semantic segmentation. The MODEL.IMAGEMASK_ON enables semantic segmentation (both training and

inference). The MODEL.ROI_IMAGEMASK_HEAD node has two configuration fields, PREDICTOR and NUM_CLASSES. We implemented the basic predictor described below. It is enabled by setting the PREDICTOR value to `MaskRCNNFPNImageMaskPredictor`. NUM_CLASSES should be set to 60 for the PASCAL in Detail dataset.

4.6. Design decisions

Our implementation is architecture-specific, it assumes that we use Mask R-CNN with Resnet with the FPN backbone. We use only the part of the FPN output with the highest resolution. We built *class ImageMaskHead* with *predictor* and *loss evaluator* modules. It's worth noticing, that semantic segmentation completely ignores the Region Proposal Network's output (technically, with only semantic segmentation enabled, we could omit region proposing, for optimization purposes).

4.6.1. Dataset preparation

We provide two functions that retrieve image masks from the dataset, called *getMulticlassMask(boxlist)* and *getCWHMulticlassMask(boxlist)*, where `boxlist` is a Mask R-CNN specific structure *class Boxlist*, defined in *root/maskrcnn-benchmark/structures/bounding_box.py*. `getMulticlassMask` returns a tensor of size (W, H). At $[i, j]$ there is an id of the class the (i, j)-th pixel belongs to. `getCWHMulticlassMask` returns tensor of size $(60, W, H)$, with 1 at position (k, i, j) if and only if the (i, j)-th pixel belongs to class k, otherwise 0 (there are 60 classes in PASCAL In Detail).

4.6.2. Predictor

Predictor is a module, that prepares feature maps to predict semantic masks. It flattens feature maps to the depth equal to the number of classes, which is a natural representation of a mask. In *predictor* obtained from the feature extractor, feature maps of the input image are convolved to reduce their dimension from 256 to NUM_CLS (number of semantic annotation classes), where 256 is a Resnet-FPN specific number, referring to output channels (in configuration file, it is MODEL.RESNETS.BACKBONE_OUT_CHANNELS value). In the PASCAL in Detail dataset case, the *predictor* output is of shape (N, 60, W, H), where N is batch size and (W, H) is input image specific - it may be (40, 24) for a (500, 300) image. It is associated with the N $>$ 1 problem - later we need to upsample it to the image size, but we need to do it one by one per image, because of different image sizes (described widely in 4.7). The workflow diagram is shown in figure 4.1.

4.6.3. Loss evaluator

When training the network, we use the loss evaluator module in each iteration to the computed value of the loss function for the backpropagation step. As a loss function we chose binary cross entropy (it is implemented in Pytorch 1.0). It cannot be applied first off, because we have the unnormalized predictions (logits) on input. Thus the softmax function per pixel is applied first, then the loss function. To obtain predictions, we interpolate each image from the batch to its initial size using nearest-neighbour interpolation. In Mask R-CNN, the loss for semantic segmentation is kept in the loss dictionary at key *loss_imagemask*.

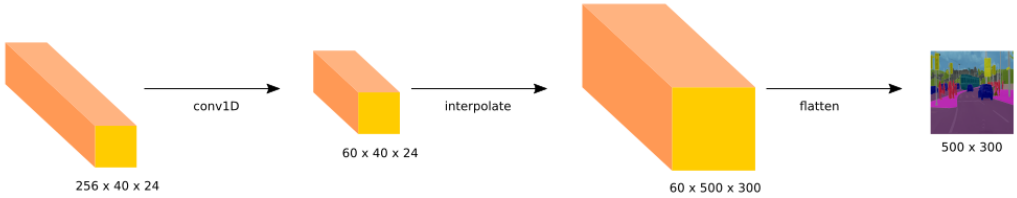


Figure 4.1: Semantic Segmentation diagram

4.7. Technical problems

The main problem in our implementation is processing multiple images at once (batch size greater than 1), because of differing image sizes. What we do is find the minimal size capable of fitting each of the images, i.a. $(600, 500)$ for images $(600, 300)$ and $(200, 500)$. We will refer to this size as (W, H) and create a tensor of size (N, W, H) (in our example it is $(2, 600, 500)$). The image is placed in the lower-left corner. Then we fill the remaining parts of the tensor with dummy values equal to 0.

4.8. Prospects for continuation

As mentioned before, most of the work has been done. However, there is still a lack of certain things, mainly evaluation. Currently, our implementation allows training the semantic segmentation task separately from other tasks and performing inference on a single image. The rest of future work concerns merging changes associated with semantic segmentation into the Mask R-CNN mainstream, to keep it compatible with the network at it's current state. It is worth noticing, that the initial implementation of Mask R-CNN did not support multi-task learning with the Detail dataset. A great effort associated with filtering annotations has been made (as mentioned in section 3.4), thus merging the semantic segmentation task may require some work in the filtering field. By adding evaluation, the accuracy of our implementation could be verified, and in case of disappointing results, the hyperparameters should be tuned to obtain the best results possible.

Chapter 5

Development & Operations

Since our work is based on provided code, we had to adjust to the prepared environment. Unfortunately, one of the requirements was a graphics processing unit (**GPU**) with CUDA support, which was a problem for us. Because of it, we decided to use remote machines designed for machine learning with multiple powerful Nvidia GPUs inside. However, this approach required us to deploy multiple times in different environments.

CUDA is a system designed by Nvidia, which allows high performance parallel computing on Nvidia GPUs. It is widely used in numerous fields of science, including machine learning [8]. Thanks to the usage of this technology, training neural networks is much faster.

5.1. Deployment

We were working on three machines in total. Two of which were provided by our client (Tensorflight) and one by Faculty of Mathematics, Informatics and Mechanics, University of Warsaw (**MIMUW**):

- kasia (Tensorflight)
  - Operating system: *Ubuntu 16.04*
  - GPUs: 4x *GeForce GTX 1080 Ti*
- bozena (Tensorflight)
  - Operating system: *Ubuntu 16.04*
  - GPUs: 4x *GeForce GTX 1080 Ti*
- sylvester (MIMUW)
  - Operating system: *Ubuntu 16.04*
  - GPUs: 4x *NVIDIA TITAN X*

It turned out that regardless of the similarities in the platforms, the deploymen t process was different and we had to approach each machine individually. We tackled different problems on each of the machines.

The process of deployment starts with installation of Anaconda – distribution of Python created especially for data science. Then, we had to link CUDA libraries to the `PATH` environment variable. After this step, the main part of deployment took place. Associated

projects and datasets had to be downloaded and set up. This was followed by installation of the required Python libraries. In the end, the network had to be built.

While working on the project, we created scripts and instructions for performing deployment. This led to a semi-automatic process with only a few human interactions.

The most difficult part was working with CUDA, especially its compiler – `nvcc`. It required us to set different paths, because the machines had different versions of the driver. On one of the machines we even had to install the CUDA driver. It appeared that Ubuntu was not entirely supported by Nvidia and a bug prevented the driver from being updated. Fortunately, we received help from our client, who managed to solve the problem.

As a huge success we can mention, that the rest of the process required only a little to none interactions. We created a `Makefile`, which automated most of the deployment. It downloads necessary code from GitHub repositories, installs required packages, downloads the datasets, and in the end builds everything. The script saved us a lot of time, as the main part of deployment was reduced from a few hours (on *kasia*) to a quarter or two (on *bozena*).

5.2. Unused machines

We also managed to deploy the network on Google Colaboratory server. Having *Tesla K80* GPU [9] it was the fastest machine we used. Unfortunately, the development of the code was inconvenient as we did not have direct access to the code and terminal. Furthermore, the session there lasts 12 hours, so we would have to deploy the network again and we would be unable to perform longer trainings. Nevertheless, deployment on Google Colaboratory required additional instructions and scripts.

What is more, we also tried to deploy the network on the computers at MIMUW. Since they contain a *Quadro P400* GPU, they are not very powerful. However, we decided to try using it during our project, since the machines were easily available. Even though, the operating system was unusual (*PLD 3.0*), the process was surprisingly smooth. Unfortunately, it appeared that the CUDA drivers were not up to date at the time and we were not able to do anything about this problem. Nevertheless, it was a valuable experience as the Linux distribution on the computer was rare and we did not have superuser permission.

5.3. Training

Training the network was a rather simple process. The provided code contained a training script. The only problem lied in understanding the configuration files of the network. Fortunately, documentation contained the most important information about the file format and the already provided configurations were of much help.

We based our configuration file on one of the configurations designed for the keypoint task for the COCO dataset. The changes we applied affected the most important parts of training, namely the number of iterations, batch size, learning rate, and image size.

We also prepared configurations for training on multiple GPUs. Unfortunately, since we were not the only users of the machines, we were not able to occupy more than one GPU for a longer period of time. Additionally, the final phase of development of our project interfered with the system that Pytorch uses for distributed training.

What is more, since we were using a pretrained network as the base of our work, we had to adjust the weights provided in the model. This resulted in creating a script, which deletes the unnecessary weights from the downloaded model. This way, we could integrate the backbone with the rest of our network.

Chapter 6

Training Results

We trained our architecture on the Detail dataset's training subset and evaluated the results on the test subset. After many days of constant training with hyperparameter fine-tuning on limited resources i.e, on only one of *kasia*'s GPUs, we managed to evaluate our net and get the following results.

6.1. Results

Below we present the results we obtained while training on 300k iterations with batch size 2 evaluated with the mean Average Precision metric.

Task	Best Result (mAP)
Boundary detection	0.239
Instance segmentation	0.195
Human keypoint detection	0.103

Table 6.1: Best results for each task

6.2. Mean Average Precision

Mean Average Precision (mAP) is an evaluation measure which computes a mean over task-specific average precisions calculated for each of the classes used in a given task. We chose to use it as it is one of the most commonly used evaluation measures in computer vision.

6.3. Interpretation

Our results are not state-of-the-art, but seeing them be much more precise than randomized predictions makes us believe that the net is able to learn specific tasks' characteristics. Because of our net's aforementioned abilities, we are convinced that future development will be able to substantially raise the quality of our net's predictions.

Due to the lack of an evaluation module for semantic segmentation, we present an example prediction in appendix B.

Chapter 7

Conclusion

7.1. What was the goal?

The goal was to fully utilize the potential of the PASCAL in Detail dataset by creating an architecture able to exploit the large variety of annotations for multi-task learning. We wanted our final implementation to have support for as many task as possible.

Fully implemented Detail dataset support would result in a highly diversified architecture with multi-task possibilities for up to 9 tasks. Such a model would be relatively unique in the field of computer vision.

7.2. What did we accomplish?

The Mask R-CNN version created by us is able to train for object detection, human keypoint estimation, instance segmentation, and semantic segmentation using the PASCAL in Detail dataset. Training results are present in the chapter 6.

Furthermore, our scripts and instructions designed for deployment can help developers work more easily with code. The deployment process is more time-efficient and less problematic.

What is more, our work enables future development and wide possibilities for researchers and developers.

7.2.1. Future possibilities of development

This project can be seen as one of the early developments of multi-task learning using the PASCAL in Detail dataset. In the future, researchers can build upon our architecture to bring support for all the possible tasks. We believe that further work on this architecture will lead to important and useful findings and scientific papers. Artificial intelligence with the capability of inference in all the computer vision problems specified in our dataset would be a step closer to nearly human vision.

Chapter 8

Work Division

- **Mateusz Biegański**
  - Contact with the client
  - Creating the semantic segmentation module
  - Deployment process automation
  - Writing configuration scripts
  - Implementing the script used to convert the PASCAL in Detail database to make it Mask R-CNN compatible
  - Implementing functions in the evaluation section
  - Presentation of the project
- **Kamil Bladoszewski**
  - Contact with the client
  - PASCAL in Detail dataset annotation and image analysis
  - PASCAL in Detail train/validation subset split
  - Preparation of configurations for single and multi GPU trainings
  - DevOps
    - Deployment of network on 3 machines
    - Deployment of network on Google Colaboratory
    - Attempt of deployment on university computer
    - Automation scripts
    - Conduction of training of the network
  - Creation of the project poster
  - Presentation of the project
- **Piotr Gierda**
  - Research regarding deep learning and multi-task learning
  - Review of scientific papers related to our project, creating summaries with useful keyword highlighting (28 papers)
  - Contact with the client

- COCO and PASCAL in Detail dataset annotations format comparison and analysis
- Implementing the script used to convert the PASCAL in Detail database to make it Mask R-CNN compatible
- Implementing the filtering module
- Integrating Mask R-CNN code with the PASCAL in Detail dataset to enable training and evaluation
- Implementing a function converting LEB128 and RLE encoded segmentation mask to polygons representation
- Implementing functions aiding semantic segmentation implementation
- Implementing the annotation filtering module for instance segmentation in evaluation section

- **Paweł Opiela**
  - Research regarding deep learning and multi-task learning
  - Review of scientific papers related to our project, creating summaries with useful keyword highlighting (28 papers)
  - Contact with the client
  - COCO and PASCAL in Detail dataset annotations format comparison and analysis
  - Implementing the script used to convert the PASCAL in Detail database to make it Mask R-CNN compatible
  - Implementing the filtering module
  - Integrating Mask R-CNN code with the PASCAL in Detail dataset to enable training and evaluation
  - Implementing the script used to select correct pretrained weights for the backbone neural net
  - Setting up optimal configuration file
  - Implementing the annotation filtering module for human keypoint detection in evaluation section
  - Leadership role

Chapter 9

Content of the CD

- Git repository with project sources
- Documentation (Bachelors.pdf)
- Presentation slides (Presentation.pdf)

9.1. Repository structure

Because we forked an already existing repository, there are some unused branches. The important ones are *master* and *dev*. The *master* branch contains code prepared to run multi-task learning with training and inference and *dev* also contains the implementation of the semantic segmentation task.

9.2. Installation

The installation guide can be found in the INSTALL.md file on the CD. After successfully following the installation guide, running `make train` trains the network on the Detail dataset training subset and then evaluates the results on the validation subset.

Appendix A

Annotation Example

The following two annotations' 'segmentation' fields are accordingly: a RLE represented mask and a polygons represented mask. This way we can clearly see the difference between these formats.


```
{
  'segmentation': {
    'size': [333, 500],
    'counts': 'eR?5X:1O00O10^la4'
  },
  'area': 27,
  'iscrowd': 1,
  'image_id': 2008000003,
  'bbox': [46, 127, 5, 6],
  'category_id': 232,
  'parts': [],
  'id': 3
}
```

```
{
  'segmentation': [[
    48.0, 132.5, 47.0, 132.5, 46.5, 132.0, 46.0, 131.5,
    45.5, 131.0, 45.5, 130.0, 45.5, 129.0, 45.5, 128.0,
    45.5, 127.0, 46.0, 126.5, 47.0, 126.5, 48.0, 126.5,
    49.0, 126.5, 50.0, 126.5, 50.5, 127.0, 50.5, 128.0,
    50.5, 129.0, 50.5, 130.0, 50.5, 131.0, 50.0, 131.5,
    49.0, 131.5, 48.5, 132.0, 48.0, 132.5
  ]],
  'area': 27,
  'iscrowd': 1,
  'image_id': 2008000003,
  'bbox': [46, 127, 5, 6],
  'category_id': 232,
  'parts': [],
  'id': 3
}
```

Example of a keypoint annotation:

```
{
  'num_keypoints': 7,
  'image_id': 2008000003,
  'bbox': [62.0, 190.0, 83.0, 243.0],
  'category_id': 284,
  'keypoints': [
    0, 0, 0, 80.9, 193.2, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 70.2,
    212.4, 1, 70.9, 226.6, 1, 67.4, 237.3, 1, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 74.8, 213.2, 1, 81.5, 226.4, 1, 80.6, 238.7, 1
  ],
  'person_id': 1
}
```

Example of a category annotation:

```
{
        'supercategory': 'animal',
        'category_id': 25,
        'name': 'bird',
        'onlysemantic': 0,
        'parts': [2, 21, 34, 38, 42, 46, 49, 59, 63, 70, 75, 78, 79, 255]
}
```

Appendix B

Result Example

Figure B.1: Demo image evaluation on trained weights

Bibliography

[1] PASCAL in Detail dataset web page https://sites.google.com/view/pasd/home

[2] PASCAL VOC web page http://host.robots.ox.ac.uk/pascal/VOC/

[3] Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John Winn, Andrew Zisserman *The PASCAL Visual Object Classes (VOC) Challenge* http://host.robots.ox.ac.uk/pascal/VOC/pubs/everingham10.pdf

[4] Iasonas Kokkinos *UberNet: Training a 'Universal' Convolutional Neural Network for Low-, Mid-, and High-Level Vision using Diverse Datasets and Limited Memory* arXiv:1609.02132v1 [cs.CV]

[5] Léon Bottou, Frank E. Curtis, Jorge Nocedal *Optimization Methods for Large-Scale Machine Learning* arXiv:1606.04838v3 [stat.ML]

[6] COCO dataset web page http://cocodataset.org/

[7] ImageNet web page http://www.image-net.org/

[8] About CUDA, Nvidia Developer https://developer.nvidia.com/about-cuda

[9] Google Colaboratory specification https://colab.research.google.com/drive/151805XTDg--dgHb3-AXJCpnWaqRhop_2

[10] Kaiming He, Georgia Gkioxari, Piotr Dollár, Ross Girshick *Mask R-CNN* arXiv:1703.06870v3 [cs.CV]

[11] Yu Zhang, Qiang Yang *A Survey on Multi-Task Learning* arXiv:1707.08114v2 [cs.LG]

[12] Jonathan Hui *Image segmentation with Mask R-CNN* https://medium.com/@jonathan_hui/image-segmentation-with-mask-r-cnn-ebe6d793272