KU LEUVEN

FACULTY OF ENGINEERING TECHNOLOGY

TECHNOLOGY CAMPUS DE NAYER

Counting shells

Are current neural networks performant enough to count various types of shells in an uncontrolled environment

Tijs VAN KAMPEN

Promotor: Prof. dr. Toon Goedemé

Masterproef ingediend tot het behalen van de graad van master of Science in de industriële wetenschappen: E-ICT Software Engineer

Academic year 2022 - 2023

Summary

Every year the Flemisch Institute for the sea organizes a shell counting day to map the diversity of our seaside. Thousands of volunteers go to the beach to count and classify shells. In this thesis, we will try to automate this process by using neural networks. The goal is to be able to count the shells in an uncontrolled environment so the volunteers would only have to take pictures of the shells and the neural network would do the rest.

This is not a trivial task, as the shells are not always in the same position, the lighting conditions are not always the same and the shells are not always the same size. The large variety of shells, with some of them being very similar, makes it even harder to detect them correctly.

We are also limited in the amount of data we can use to train our neural network as we do not have a large annotated dataset of shells. We will thus have to use a few-shot approach to train our network.

Abstract

In this thesis, we will explore the field of few-shot object detection to find out if recent advancements in the field have made it performant enough to be able to detect and count shells on a beach.

Few-shot object detection is a newer field of research in computer vision that has been gaining traction in the last few years. As opposed to the related field of image classification, where the goal is to classify an image into a class, object detection aims to detect and classify objects in an image. This is a more complicated task, as the network has to not only classify the objects but also localize the objects in the image. The added limitation of not having a lot of images of the objects to work with results in not having sufficient information on the objects to detect them. The only way to achieve reliable detection is to make up for the gap in our knowledge in a different way. This is where the field of few-shot object detection comes in. It allows us to first gain generic knowledge by learning from a large dataset and then fine-tuning that knowledge to our specific task by learning from a small dataset.

Keywords: Few-shot object detection, object detection, neural networks, shells, beach, computer vision, machine learning

Contents

Chapter 1

Introduction

Once per year, nearly a thousand volunteers travel to the Belgian coast to collect and categorize the shells that wash up on the beach. This data is collected by the Flemish Institute For The Sea (VLIZ) to study populations of marine mollusks and the impact of their environment (climate change, fishing, etc) on the population. The volunteers participating in this study are mostly enthusiasts, but also scientists and families with children. To ensure a good quality of the data, most volunteers participate in a workshop prior to the activity. The counting of the shells is done by walking along the beach and log every individual shell that is found. This is a very time-consuming process, and the volunteers are often not very experienced in counting, resulting in mistakes with all but the most common shells. When a volunteer finds a shell that they are not familiar with, they can contact a helpdesk to help them. The flowchart of the current process can be found in figure 1.1.

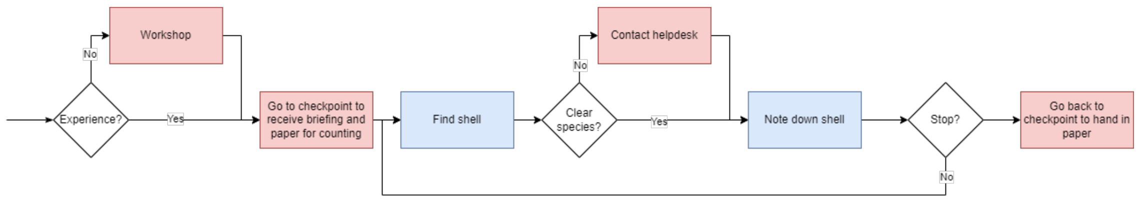

Figure 1.1: The current process of collecting data.
Marked blue is the volunteer's actions, and marked red are the parts that involve experts.

The fact that the project relies on volunteers to do most of the legwork, combined with experts having to man the checkpoints and the helpdesk, makes the project unscalable beyond having a single dedicated day each year. With over 5 million people visiting the Belgian coast every year[9], there is a lot of potential data to collect if the process of collecting the data could be simplified and accessible to anyone visiting the beach at any time.

In this thesis, we will attempt to simplify the process of data collection so that it can be done by anyone, anywhere, at any time. We will do this by training a counting network so that shells can be recognized in an image and counted automatically. This is already done on a smaller scale by VLIZ with Obsidentify, a mobile app and website where users can submit pictures of a single shell and get a result of what kind of shell it is. This is a useful tool, but taking a close-up picture of every single shell is again a very time-consuming process.

As no dataset exists with large quantities of annotated pictures of beaches, we will have to work with a dataset of limited size to train the neural network. After the successful completion of this thesis, the new ideal scenario for collecting data can be found in figure 1.2. Compared to the current process, found in figure 1.1, this new process nearly eliminates the experts' involvement and thus makes the process scalable.

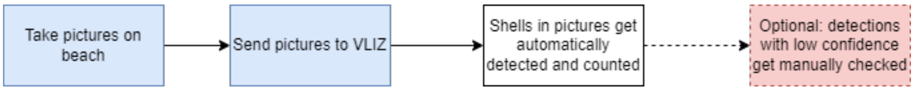

Figure 1.2: The new process of collecting data.

We will be studying if current counting networks are performant enough to recognize shells in beach images. We will build up to this by first training a network to count objects from a more established dataset in order to have a baseline to compare our model to. Afterward, we will then train that network with a small dataset to count shells and study its performance.

In the remainder of this thesis, we will first discuss the state of the art in the field of object detection and counting, with a focus on few-shot learning. We will then discuss the datasets and the network architecture that we will be using. In the second semester, we will implement the network and train it on the datasets. We will then discuss the results and the limitations of our model. Finally, we will discuss future work that can be done to improve the model.

Chapter 2

Literature Review

In this chapter, we will review the state of the art in the field of object counting. We will study the techniques commonly used for counting and go in more depth about the topic of few-shot object detection and why it should be applied to our problem. Finally, we will discuss the metrics used to evaluate the performance of the models.

2.1 Crowd Counting

Counting networks are an established concept in machine learning as numerous papers tackle the issue of counting humans, cars, animals or cells. What those have in common is that they only encompass a small set of possible categories to count and that, as they have a large real-life use, large annotated datasets exist like ShanghaiTech[22] and COWC[14]. The problem we are trying to solve is a bit different as we want to count a large set of objects and yet we don't have a large dataset to train on.

The methodology behind heuristic counting networks has three big streams[6]. The first applies a detection method to the image and then counts the number of detected objects. Many different detection methods can be used, from looking for characteristic features to matching the shape of the objects. The second takes a more global approach by first extracting features, textures, gradients and other information from the image as a whole and then using those to count the objects. The third method is not used on static images, but on video. It assumes that the objects are moving in clusters and uses that to predict the movement of the objects and improve detection.

Out of those three methods, the third one is not applicable to our problem as we are trying to detect unmoving objects in a still image. Both the first and second methods are applicable to our problem, however, both have the problem of requiring a large dataset to train on. We will have to use a method that doesn't require a large dataset, which is where few-shot learning comes in. In the domain of few-shot learning the first method, object detection, is the most common. In the next section, we will go more in-depth about few-shot object detection.

2.2 Object detection

The history of techniques used for object detection can be split into two parts: traditional methods and deep learning-based methods. Before 2012 the traditional methods were the most common, as hardware was not yet powerful enough to train deep learning models. After 2012, with hardware becoming more powerful, deep learning-based methods became more common. The deep learning-based methods can be split into two categories: single-stage and two-stage methods. Two-stage methods split the problem into two stages, first detecting the objects and then classifying them and are thus more accurate but slower than single-stage detectors. In this section, we will go over some of the major milestones in the two-stage detector branch of the deep learning part of the history of object detection as for our problem accuracy is the highest priority.

2.2.1 R-CNN[5]

R-CCN, shown in fig 2.1, was the first two-stage deep learning method. It obtains region proposals using a method to obtain category-independent region proposals, like selective search[18], and then classifies those regions using a CNN. While it obtained a large improvement over the SOTA, it was quite slow as with each image it had to run the CNN on 2000 regions to extract features. Many of those 2000 would be redundant as they overlap.

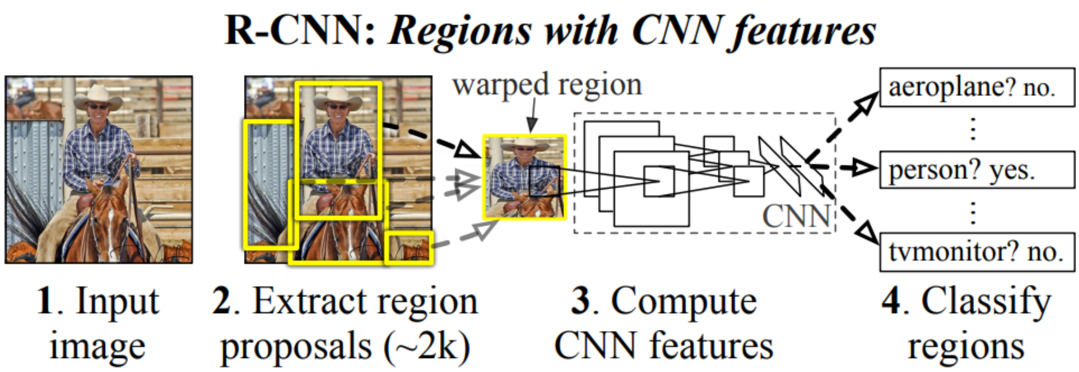

Figure 2.1: RCNN architecture. Image from Girshick et al. [5].

2.2.2 SPPNet[7]

SPPNet, shown in fig 2.2, takes a different approach than R-CNN. Conventional CNN-based methods are limited by the fact that the fully connected layers at the end of the network require a fixed input size. This means that the input image has to be resized to a fixed size for each region, which leads to loss of information and/or bad representation, this can be seen in 2.1. SPPNet solves this problem by adding a spatial pyramid pooling layer (SPP) after the last convolutional layer, a comparison with conventional CCNs can be found in fig 2.3. Convolutional layers can take any size input, which is then pooled to the fixed size required for the fully connected layers by the SPP layer. As the convolutional layers are only run once per image, this greatly improves the speed of the network, reaching 20-100x speedup compared to R-CNN. The SPP layer consists of parallel max-pooling layers with differing amounts of pooling regions, which are then concatenated to form a fixed-length representation, as shown in fig 2.2.

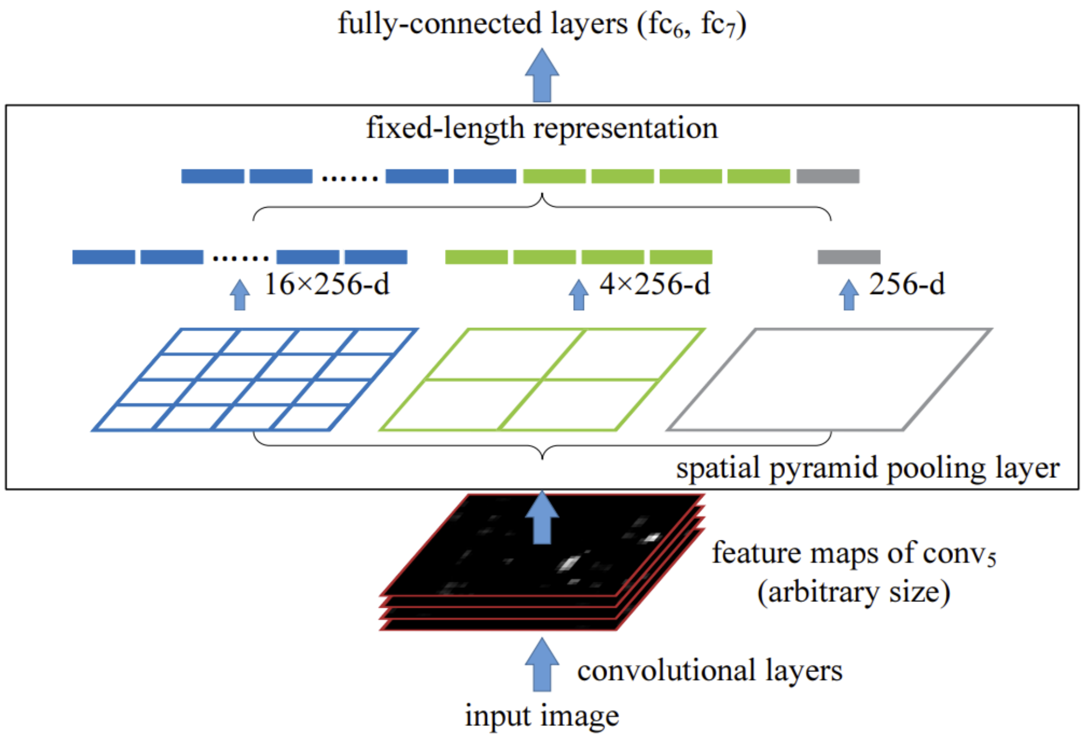

Figure 2.2: SPPNet architecture. Image from He et al. [7].

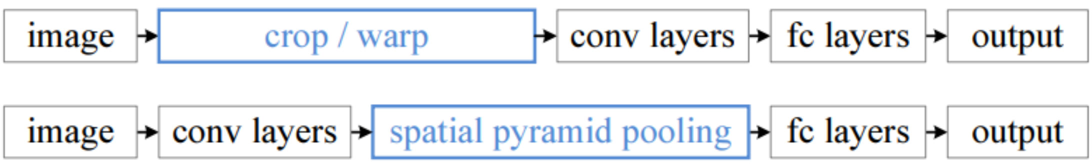

Figure 2.3: Conventional CNN vs SPPNet. Image from He et al. [7].

2.2.3 Fast R-CNN[4]

Fast R-CNN, shown in fig 2.4, combines the region proposals of R-CNN with the method of SPPNet. It runs the CNN on the whole image and then applies max pooling for each region proposal, they call this region of interest pooling (RoI pooling). The RoI layer is a special case of the SPP layer, the case where the number of pooling regions is 1. As the time-consuming CNN is only run once per image, the training and inference time is greatly improved, while the RoI pooling layer increases the accuracy of the network.

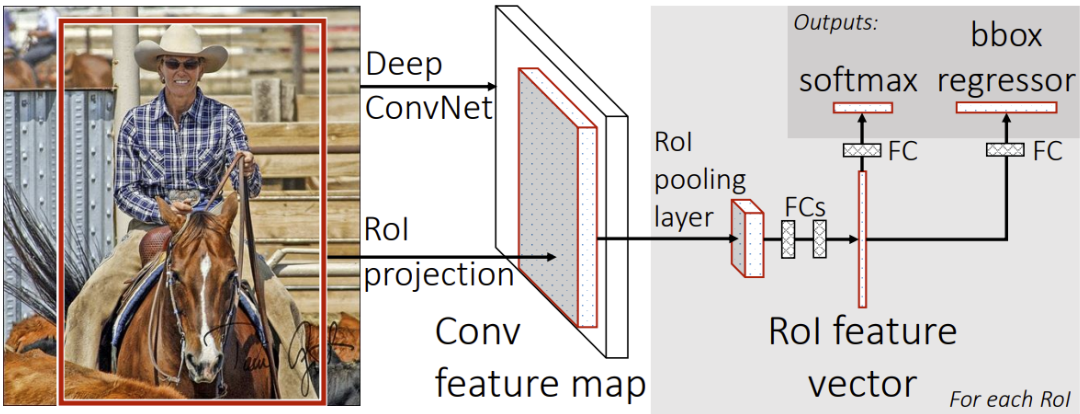

Figure 2.4: Fast R-CNN architecture. Image from Girshick [4].

2.2.4 Faster R-CNN[16]

Faster R-CNN, shown in fig 2.5, improves on Fast R-CNN by replacing the region proposal method with a region proposal network (RPN). The RPN is a fully convolutional network that takes the feature map from the CNN and outputs a set of rectangular object proposals, each with an objectness score. The RPN is trained end-to-end with the rest of the network, which allows the network to learn the region proposal method that works best for the task at hand. This greatly improves the speed of the network, as the RPN is much faster than the region proposal method used in Fast R-CNN.

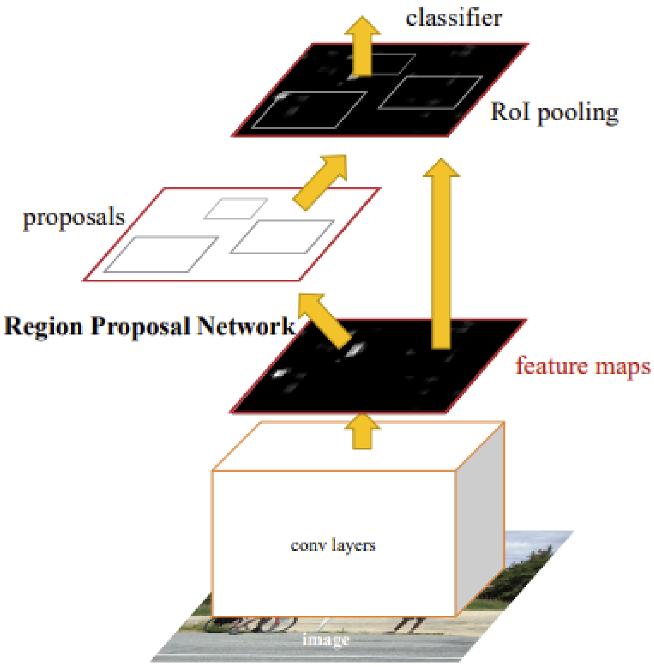

Figure 2.5: Faster R-CNN architecture. Image from Ren et al. [16].

2.3 Vision Transformers

This section will be about the workings of vision transformers

2.4 Few-shot object detection

Few-shot object detection is a technique that has been gaining popularity in the last few years, but interest in training a neural network to classify without a big annotated dataset appeared as early as 2008 with zero-shot learning in Chang et al. [2]. It allows us to train a model with few annotated images, which is useful in scenarios where it isn't possible to get a large annotated dataset. Few-shot attempts to mirror the way humans learn. During our life we come across many new objects and we are able to recognize them even though we only saw them a few times. We do this by drawing on our knowledge of other objects and using that to recognize the new object[1].

Different approaches to few-shot learning vary on a few characteristics

- The type of architecture used
- The amount and type of data used

In this section, we will go over the different options for each of these characteristics.

2.4.1 Method

There are two different methodologies that can be applied to few-shot learning, transfer learning and meta-learning. Each of these has its own advantages and disadvantages.

Transfer learning

Transfer learning is a technique that has been used for a long time in machine learning. It allows us to use a model that has been trained on a large dataset as a base and, with a few changes to mitigate the small size of the novel dataset in few-shot learning, finetune (the last layers) on a novel dataset. The advantages of this method are that it is relatively easy to implement and it is fast. One of the problems with this method is that, because of the small size of the novel dataset, the Region proposal network (RPN), which provides class-agnostic bounding boxes, can not be properly trained and can sometimes completely miss the novel object classes. Mitigations for this problem do, however, exist. [21, 19, 3, 17, 20].

Meta-learning

Meta-learning learns on a higher order of abstraction. Instead of learning how to detect objects it learns how to learn to detect objects. It does this with the help of a large dataset, by learning how to best extract and differentiate the features of its classes. Due to the dataset being large, this can then be applied to a novel dataset. It is best if the novel dataset is similar to the large dataset, as it will be able to generalize better. Practically meta-learning is most commonly done by introducing a support branch[8], displayed in figure 2.6. An advantage of this method is that it is better at the detection of alike novel classes due to the meta loss, a loss function on the support branch. The disadvantages are its complexity and that it is slower to train than transfer learning due to the aforementioned support branch. As the support branch is computed once after training, it is not significantly slower during inference.

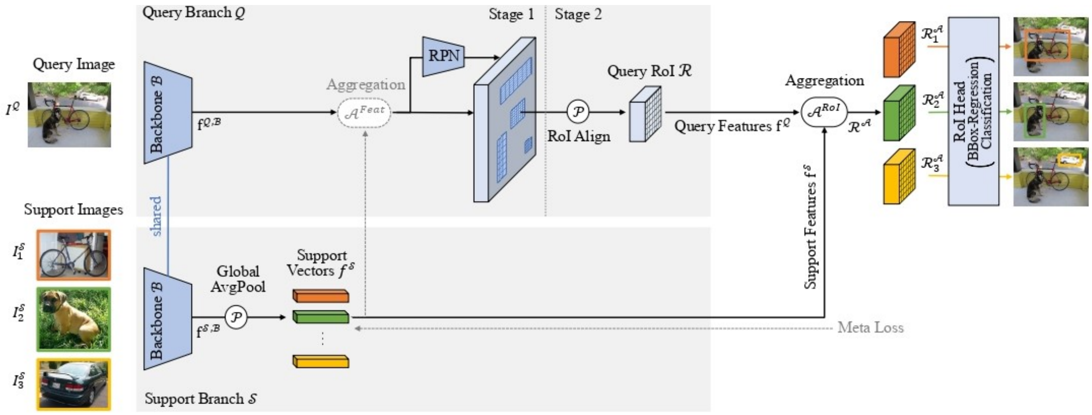

Figure 2.6: Dual branch meta learning. Image from Köhler et al. [8].

2.4.2 Data

The amount and type of data used in few-shot learning is also an important factor. A common way to describe a few shot task is ”N-Way, K-Shot”. Where N is the number of classes and K is the number of examples per class. The more examples we have per class, the easier it is to learn. The larger the number of classes the harder. Models are often benchmarked with increasing K to see how they perform with values of K often set at (2,) 5, 10 and 30. When the amount of images decreases even further we enter a whole new category of few-shot learning, one-shot learning and zero-shot learning. Finally, the requirements for the type of annotations on the data can vary depending on the training method used. Supervised requires a fully annotated dataset, semi-supervised a partially annotated dataset and unsupervised doesn’t need labels at all.

One-shot learning

While one-shot learning is not a new concept, applying it to object detection is hard. Early applications used a siamese backbone, as is often seen in other one-shot applications, but this was not very successful [12]. However, recently OWL-ViT [13] improved one-shot object detection by a large margin, reaching 41.8 mAP for one-shot object detection on the COCO dataset. OWL-Vit will be discussed in more detail in section 2.6.

2.5 Metrics

In machine learning, it is important to test the model after training, to evaluate its performance. To test the model's accuracy a part of the initial dataset is split off into a test set and never used when training. As we have the ground truth for the test set we can compare it with the network output to find if the detections are correct.

To find if the model output matches the expected output we use the Intersection over Union (IoU) metric. This compares the area of the input and output bounding boxes for each detection by dividing the area of intersection by the area of union. If the IoU, calculated as shown in 2.1, is above a threshold (th) it is considered a detection.

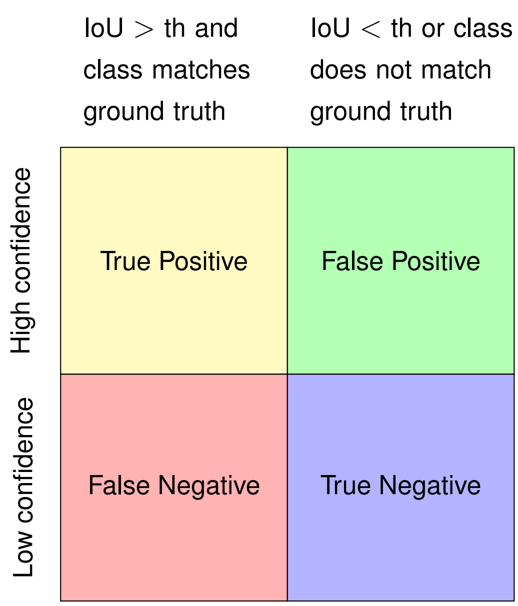

Figure 2.7: IoU and class match to find the type of detection.

$$\text{IoU} = \frac{\text{area of intersection}}{\text{area of union}} \tag{2.1}$$

Each detection can be put into one of four categories, based on if and how well it matches the ground truth, listed below.

- True positive: The model correctly detects an object and the IoU is above the threshold.
- False positive: The model detects an object but the IoU is below the threshold or the model mislabels the object.
- False negative: The model does not detect an object but it should have.
- True negative: The model does not detect an object and it should not have, this is not used in the metrics as it is not very useful.

Using this a few key metrics can be calculated. The main metrics we will use are precision, recall and their derivatives. Precision (2.2) is the ratio of true positives to the total number of positives. Recall (2.3) is the ratio of true positives to the total number of detectable positives.

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives} \tag{2.2}$$

$$Recall = \frac{TruePositives}{TruePositives + FalseNegatives} \tag{2.3}$$

The model assigns each detection a confidence score, the dividing threshold between high and low confidence can be chosen freely. A high confidence threshold will result in a low recall but high precision. A low confidence threshold will result in a high recall but low precision. Plotting the precision and recall against the threshold results in a precision-recall curve.

Averaging the precision across all recall levels results in the average precision (AP) metric. Averaging the AP over all classes results in the mean average precision (mAP) metric. The mAP is the most common metric used to evaluate object detection models.

2.6 State of the art

In this section, we will go over the state of the art in few-shot object detection with a focus on the previously discussed methods and types of data.

Simple Open-Vocabulary Object Detection with Vision Transformers. (Minderer et al. [13])

Minderer et al. [13] introduce a new method, Vision Transformer for Open-World Localization, or OWL-ViT. It uses recent developments in language encoders and contrastive image-text training, using both positive and negative examples of image-text pairs to divide the feature space. These models are trained on loosely matched image-text pairs, which can be abundantly found on the internet. They start from the Vision Transformer (ViT) architecture and pre-train it on a large dataset of these image-text pairs. The token pooling layer downsamples the output embeddings to optimize computation, for open-vocabulary detection this does not work, however. The token pooling layer is thus replaced with a set of lightweight classification and box heads at each output token. The classification head does a linear projection, while the box head is a simple MLP with one hidden layer with a gelu activation function. The whole model (both text and image encoders) is then finetuned on standard object detection datasets. It is then ready to be used for one-shot object detection as it can take text, but also image-derived embeddings to find matching objects in the query image.

Integrally Migrating Pre-trained Transformer Encoder-decoders for Visual Object Detection. (Liu et al. [11])

Liu et al. [11] extend the ViT architecture by making full use of the transformer encoder-decoder instead of merely using the encoder as its backbone, as many other methods do. Whereas other

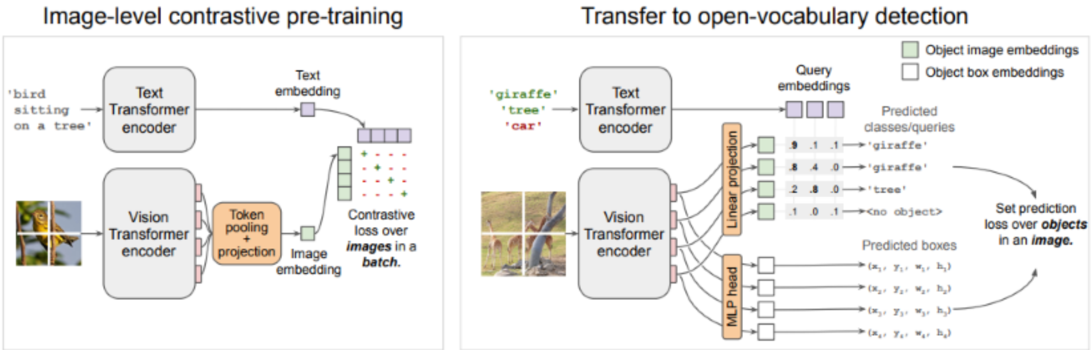

Figure 2.8: Original ViT to OWL-ViT. Image from Minderer et al. [13].

methods use the encoder as their backbone and initialize the FPN, RPN and detector head from scratch, imTED instead uses the decoder as its detector head. As such the only parts that need to be trained are the FPN, RPN and output layers of the FPN and decoder. This reduces the number of parameters that are randomly initialized by 81.3%.

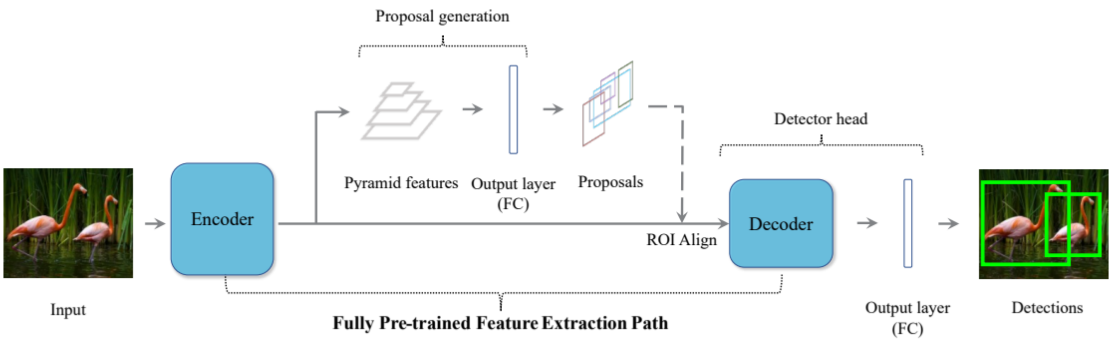

Figure 2.9: imTED architecture. Image from Liu et al. [11].

Hierarchical Attention Network for Few-Shot Object Detection via Meta-Contrastive Learning. (Park and Lee [15])

Park and Lee [15] expand upon Faster R-CNN [16] by introducing a hierarchical attention module (HAM) and a meta-contrastive learning module (Meta-CLM). The HAM combines the robustness of global attention with the local context information of a convolutional network by first applying local attention and then applying global attention. The stages can be found in fig 2.10. The Meta-CLM combines contrastive learning with meta-learning. It works on the same principles as contrastive learning, only instead of positive and negative images it uses positive and negative image pairs. Due to being metric-based, it can be used without any finetuning on novel classes.

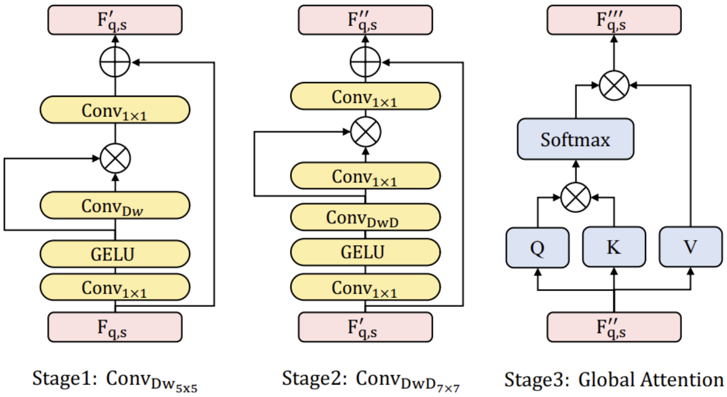

Figure 2.10: Detailed view of the HAM. Image from Park and Lee [15].

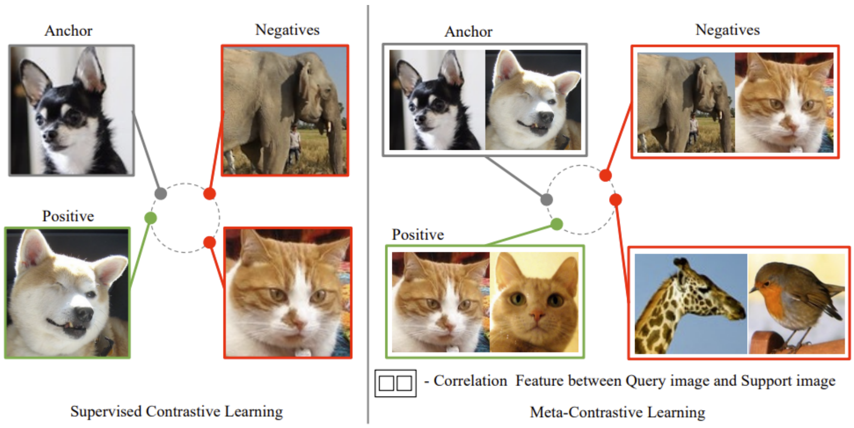

Figure 2.11: Meta-contrastive learning. Image from Park and Lee [15].

2.7 Conclusion

	Scores on COCO	Special characteristic
OWL-ViT	49.1* [1 shot]	1-Shot-N-Way without retraining
	41.8* [10 shot]	
	/	
imTED	/	Reduced training time/diffucულty
	22.5 [10 shot]	
	30.2 [30 shot]	
hANMCL	13.4 [1 shot]	1-Shot-N-Way without retraining
	22.4 [10 shot]	Faster R-CNN based, so less performant hardware required
	25.0 [30 shot]	Meta-learning suited to divide metric space of our alike shells

*Results on AP50

In this chapter we have studied object counting, narrowing it down to few-shot object detection to then count the detections. We went into more detail regarding the different ways to implement few-shot learning to detect objects. Studying the SOTA methods for few-shot object detection, we found that a wide variety of models, each with different advantages and disadvantages exist. OWL-ViT and hANMCL are capable of few-shot object detection without finetuning/retraining, thus they are a good fit to establish a baseline. In terms of viability, hANMCL is faster R-CNN-based and uses a convolutional backbone, as opposed to OWL-ViT and imTED which use ViT, a transformer based backbone. This results in lower requirements in terms of hardware. We will start with finetuning hANMCL, if we have the time and hardware to do so we will also finetune OWL-ViT and imTED.

2.8 Planning

4 weeks left:

- Establising baseline with the one-shot no retraining models & Annotating the dataset
- Training the models
- Evaluating the models and writing results
- Final touches I guess
-
- All weeks: correcting mistakes
- (Week 5: acknowledge that 4 weeks probably wasn't enough)

Chapter 3

Implementation

In this chapter, we will discuss the implementation of the network. We will first discuss the dataset and network architecture we will be using. We will then go into detail about the implementation of the network and the training process.

3.1 Dataset

In this section, we will go over the datasets used in this paper, with a focus on the shell dataset we are introducing ourselves. With this information, we can better choose candidate models for our task.

3.1.1 COCO

Microsoft's Common Objects in Context (COCO) dataset is a large-scale object detection and segmentation dataset. It contains 330K images with 1.5M instance annotations of 80 different classes. It is split into a training set of 118K images, a validation set of 5K images and a test set of 40K images(the other images are unlabeled). Lin et al. [10]

3.1.2 Shells

The shell dataset is a new dataset we are introducing ourselves. As of this writing, it contains 300 images of shells, with 0 annotations. The images shot taken with a cellphone camera on the Belgian coast. Depending on the chosen model, either all images will be annotated or the dataset can be expanded with more images.

This section will be expanded on in the future when annotations are made and the dataset is possibly expanded.

Bibliography

[1] Irving Biederman. Recognition-by-components: a theory of human image understanding. *Psychological review*, 94(2):115–147, 1987. URL https://doi.org/10.1037/0033-295X.94.2.115.

[2] M-W Chang, L Ratinov, D Roth, and V Srikumar. Importance of semantic representation: Dataless classification. *Importance of semantic representation: Dataless classification*, 2008. URL https://www.aaai.org/Library/AAAI/2008/aaai08-132.php.

[3] Zhibo Fan, Yuchen Ma, Zeming Li, and Jian Sun. Generalized few-shot object detection without forgetting. *CoRR*, abs/2105.09491, 2021. URL https://arxiv.org/abs/2105.09491.

[4] Ross B. Girshick. Fast R-CNN. *CoRR*, abs/1504.08083, 2015. URL http://arxiv.org/abs/1504.08083.

[5] Ross B. Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, abs/1311.2524, 2013. URL http://arxiv.org/abs/1311.2524.

[6] Khouloud Ben Ali Hassen, José J. M. Machado, and João Manuel R. S. Tavares. Convolutional neural networks and heuristic methods for crowd counting: A systematic review. *Sensors*, 22 (14), 2022. ISSN 1424-8220. doi: 10.3390/s22145286. URL https://www.mdpi.com/1424-8220/22/14/5286.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *CoRR*, abs/1406.4729, 2014. URL http://arxiv.org/abs/1406.4729.

[8] Mona Köhler, Markus Eisenbach, and Horst-Michael Gross. Few-shot object detection: A survey. *CoRR*, abs/2112.11699, 2021. URL https://arxiv.org/abs/2112.11699.

[9] Kustportaal. Toerisme en recreatie, 2021. URL https://www.kustportaal.be/nl/toerisme-en-recreatie#:~:text=De%20Belgische%20kust%20is%20de,in%20totaal%2027.723.420%20overnachtingen. Accessed 8 Jan 2023.

[10] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2015.

[11] Feng Liu, Xiaosong Zhang, Zhiliang Peng, Zonghao Guo, Fang Wan, Xiangyang Ji, and Qixiang Ye. Integrally migrating pre-trained transformer encoder-decoders for visual object detection, 2022.

[12] Claudio Michaelis, Ivan Ustyuzhaninov, Matthias Bethge, and Alexander S. Ecker. One-shot instance segmentation. *CoRR*, abs/1811.11507, 2018. URL http://arxiv.org/abs/1811.11507.

[13] Matthias Minderer, Alexey Gritsenko, Austin Stone, Maxim Neumann, Dirk Weissenborn, Alexey Dosovitskiy, Aravindh Mahendran, Anurag Arnab, Mostafa Dehghani, Zhuoran Shen, Xiao Wang, Xiaohua Zhai, Thomas Kipf, and Neil Houlsby. Simple open-vocabulary object detection with vision transformers, 2022.

[14] T. Nathan Mundhenk, Goran Konjevod, Wesam A. Sakla, and Kofi Boakye. A large contextual dataset for classification, detection and counting of cars with deep learning. *CoRR*, abs/1609.04453, 2016. URL http://arxiv.org/abs/1609.04453.

[15] Dongwoo Park and Jong-Min Lee. Hierarchical attention network for few-shot object detection via meta-contrastive learning, 2022.

[16] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015. URL http://arxiv.org/abs/1506.01497.

[17] Bo Sun, Banghuai Li, Shengcai Cai, Ye Yuan, and Chi Zhang. FSCE: few-shot object detection via contrastive proposal encoding. *CoRR*, abs/2103.05950, 2021. URL https://arxiv.org/abs/2103.05950.

[18] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders. Selective search for object recognition. *International Journal of Computer Vision*, 104(2):154–171, 2013. URL https://ivi.fnwi.uva.nl/isis/publications/2013/UijlingsIJCV2013.

[19] Anh-Khoa Nguyen Vu, Nhat-Duy Nguyen, Khanh-Duy Nguyen, Vinh-Tiep Nguyen, Thanh Duc Ngo, Thanh-Toan Do, and Tam V. Nguyen. Few-shot object detection via baby learning. *Image and Vision Computing*, 120:104398, 2022. ISSN 0262-8856. doi: https://doi.org/10.1016/j.imavis.2022.104398. URL https://www.sciencedirect.com/science/article/pii/S0262885622000270.

[20] Yan Wang, Chaofei Xu, Cuiwei Liu, and Zhaokui Li. Context information refinement for few-shot object detection in remote sensing images. *Remote Sensing*, 14(14), 2022. ISSN 2072-4292. doi: 10.3390/rs14143255. URL https://www.mdpi.com/2072-4292/14/14/3255.

[21] Weilin Zhang, Yu-Xiong Wang, and David A. Forsyth. Cooperating rpn's improve few-shot object detection. *CoRR*, abs/2011.10142, 2020. URL https://arxiv.org/abs/2011.10142.

[22] Yingying Zhang, Desen Zhou, Siqin Chen, Shenghua Gao, and Yi Ma. Single-image crowd counting via multi-column convolutional neural network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

FACULTY OF ENGINEERING TECHNOLOGY
TECHNOLOGY CAMPUS DE NAYER
Jan De Nayerlaan 5
2860 SINT-KATELIJNE-WAVER, Belgium
tel. + 32 15 31 69 44
fet.denayer@kuleuven.be
www.fet.kuleuven.be