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 of 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 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 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 one of a few classes, object detection aims to detect objects from a few classes in an image. This is a more difficult task, as the network has to not only classify the image but also localize the objects in the image. The added restriction of few-shot, not having a lot of images of the objects we want to detect to work with, makes it so that we do not have 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

Every year, one day 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. The counting of the shells is done by walking along the beach and noting every shell that is found individually. 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.

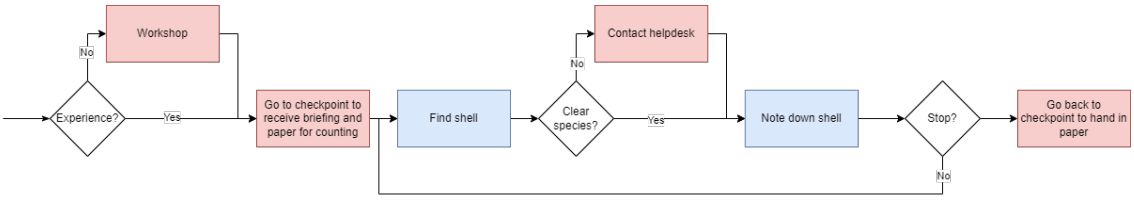

Figure 1.1: The current process of collecting data.
Marked blue is the volunteer's actions, and marked red is the parts that involve experts.

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

In this thesis, we will attempt to simplify the process of collecting data so that it can be done by anyone, anywhere, at any time. We will do this by training a counting network to recognize shells in an image and count them automatically. This is already done on a smaller scale by VLIZ with obsidentify. Obsidentify is 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 a very time-consuming process. We will have to work with a limited dataset to train the neural network as no dataset exists with large quantities of annotated pictures of beaches. 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.

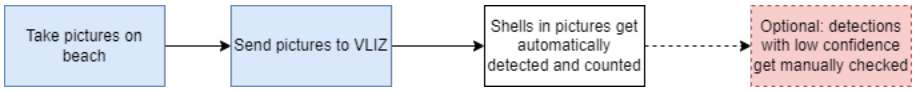

Figure 1.2: The new process of collecting data.

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 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 the future work that can be done to improve the model.

Chapter 2

Literature Review

In this chapter, we will go over the state of the art in the field of object counting. We will go over 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 quite 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 for these problems like ShanghaiTech and COWC. The problem we are trying to solve is a bit different as we want to count a large set of objects and 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 to train on, 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 Few-shot object detection

Few-shot object detection is a technique that has been gaining popularity in the last few years. It allows you to train a model on a small dataset, which is useful in scenarios where it isn't possible to get a large dataset to train due to the cost or the time it would take to get it. Interest in making a model to classify text without a big annotated dataset appeared as early as 2008 with zero-shot learning in Chang et al. [2]. The method we will be using, few-shot, differs from zero-shot as we have a few images to train on. The idea behind few-shot is as follows if a human can recognize an object after seeing it a few times, then a machine should be able to do the same. Humans achieve this because they have seen many types of objects and can use that knowledge to extract significant features of a new object[1]. In essence, few-shot learning does the same. It relies on a large dataset of (similar) objects and then applies the knowledge gained from that to a new object, with fewer examples.

Practically this can be done in two different ways. The first way is taking a pre-trained model and using transfer learning to finetune it on the new small dataset. The second way is meta-learning, where the model learns to learn. This means that the model will learn how it should learn to recognize new objects from a large dataset to then apply that to the small dataset. Both of these methods have their characteristics, which we will go over in the next section.

2.2.1 Transfer learning

Transfer learning is a technique that has been used for a long time in machine learning. In few-shot 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, finetune it on a novel dataset. A prevalent base model is Faster R-CNN[10], which improves on Fast R-CNN[5] by introducing attention mechanisms in the form of a Region Proposal Network (RPN) to provide guidance.

Using transfer learning for few shot object detecting has multiple areas that can and have been improved to increase performance. The largest area is the RPN missing the novel object classes [16, 12, 4, 11, 13].

2.2.2 Meta-learning

Meta-learning is a more recent technique in few-shot detection. The idea behind meta-learning is that the model learns how to learn to detect new objects. The are two different meta-learning principles, single-branch and dual-branch meta-learning. As dual branch meta-learning, displayed in figure 2.1, is generally more effective, we will focus on that. Many approaches build on top of Faster R-CNN with a ResNet backbone.[8]

Dual branch meta-learning trains using two branches, a query branch and a support branch. The query branch contains full-size images, while the support set contains cropped images of a single labeled object. Both branches try to extract relevant features from their respective images with a shared backbone. The features from the query branch are processed by an RPN and an RoI

align resulting in the query RoIs. The features from the support branch are pooled through global average pooling resulting in a support feature vector. Each support feature vector encodes class-specific information which is used to guide the ROI head to recognize objects of the same class. Aggregating the query RoI with the support feature vector results in category-specific RoIs which are then fed into a shared RoI head for bounding box regression and classification. Because the RoI head is shared, it must generalize across all classes. This is what happens during training, during inference the support branch is no longer needed as the support features for the novel classes are computed once.

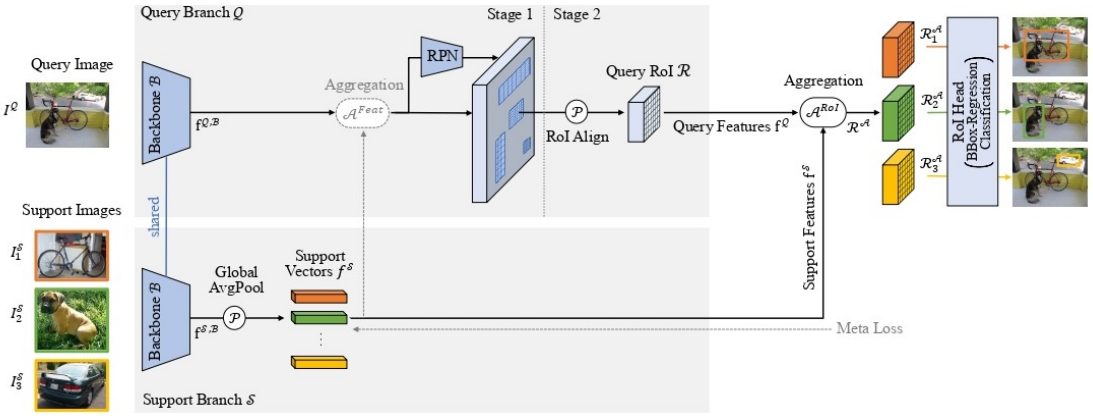

Figure 2.1: Dual branch meta learning. Image from Köhler et al. [8].

2.3 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 never used when training. As the unseen inputs are annotated we can find how close the output is to the expected output. With this few key metrics can be calculated. The main metrics we will use are precision, recall and their derivatives. Precision (2.1) is the ratio of true positives to the total number of positives. Recall (2.2) is the ratio of true positives to the total number of detectable positives.

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives} \tag{2.1}$$

$$Recall = \frac{TruePositives}{TruePositives + FalseNegatives} \tag{2.2}$$

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 2.3, is above a certain threshold it is considered a detection.

$$\text{IoU} = \frac{\text{area of intersection}}{\text{area of union}} \tag{2.3}$$

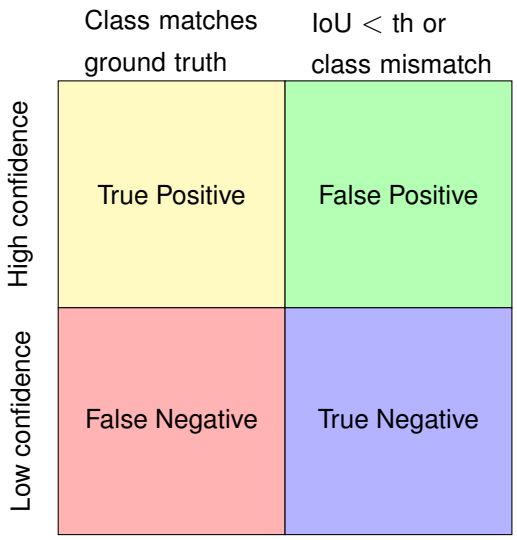

Figure 2.2: IoU and class match to find the type of detection.

Each detection is then evaluated based on how confident the model is and if the detected class matches the ground truth. Below is a list of all the different types of detections:

- 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.

The threshold between high and low confidence can be varied, a high threshold will result in a low recall but high precision. A low 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.

2.4 State of the art

In this section, we will go over the state of the art in few-shot object detection with both transfer learning and meta-learning.

2.4.1 Transfer learning

Transfer learning, as previously discussed, takes a model trained on a large dataset and finetunes it on a novel dataset. In this section, we will go over the state of the art in few-shot object detection

with transfer learning.

2.4.1.1 A Low-Shot Transfer Detector for Object Detection(Chen et al. [3])

Chen et al. [3] proposes a new architecture, designed to mitigate transfer learning difficulties in few-shot detection by combining Single Shot Detector (SSD) and Faster R-CNN into Low-Shot Transfer Detector (LSTD). The LSTD architecture is shown in figure 2.3. In LSTD bounding box regression is first performed according to SSD. The advantage of using SSD is that all classes share a single bounding box regressor, making it suitable for pretraining on a large dataset. The object detection is performed according to a modified Faster R-CNN. Faster R-CNN is modified, replacing the fully connected layers with two convolutional layers, to reduce overfitting.

On the ImageNet2015 dataset, it outperforms Faster R-CNN and SSD. On the VOC2007 and the VOC2010 dataset, it outperforms weakly and semi-supervised object detection methods when the number of training images is above 1. Note that both the weakly and the semi-supervised approaches require the full training set, so LSTD outperforms them with merely 0.4% training data.

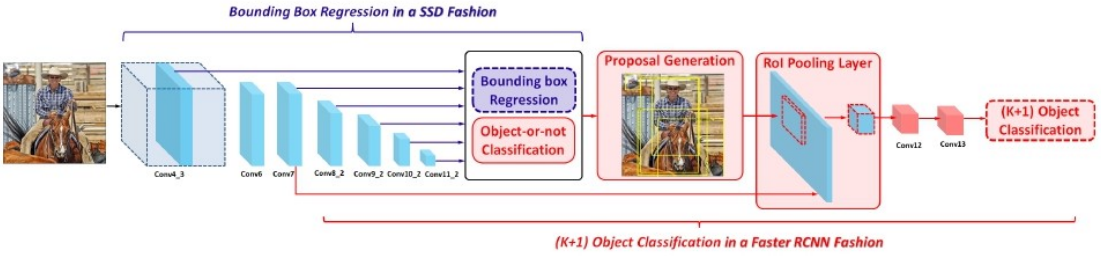

Figure 2.3: LSTD architecture. Image from Chen et al. [3].

2.4.1.2 DeFRCN: Decoupled Faster R-CNN for Few-Shot Object Detection (Qiao et al. [9])

Qiao et al. [9] extends Faster R-CNN[10] by introducing Gradient Decoupled Layers (GDL) and Prototypical Calibration Block to create Decoupled Faster R-CNN (DeFRCN). The DeFRCN architecture is shown in figure 2.4.

During forward propagation, the GDLs apply an affine transformation layer to enhance feature representation and perform forward decoupling. During backward propagation, the GDL takes the gradient from the subsequent layer, multiplies it with a constant and passes it to the preceding layer. This reduction ensures that the update speed of the backbone is slower than the update speed of the connected network, this prevents overfitting due to the small dataset. In practice, the gradient between the backbone and the RPN is set to 0 as the RPN is largely class-agnostic, and the gradient between the RPN and the RCNN Head is set to 0.75 during base training and 0.01 during fine-tuning.

The Prototypical Calibration Block (PCB) is a metric-based score refinement module. It aims to eliminate high-scored false positives and remedy low-scored massing samples. It consists of a

strong classifier from an ImageNet pre-trained model, an RoI-align layer and a prototype bank. In practice it first extracts the original image feature maps, it then employs RoI-align with ground-truth boxes to produce instance representations. Based on these features the support set is shrunk to a prototype bank. When given an object proposal from the fine-tuned few-shot detector, it first performs RoI-align on the predicted box to generate the object features. It then calculates the cosine similarity between the object features and the prototype bank. In the end, a weighted aggregation between the original score and the cosine similarity is performed. The weight is fixed to 0.5 in practice, resulting in equal weight between the original score and the cosine similarity. As the PCB module is offline and does not require any additional training, it can be easily integrated into any existing object detection framework.

On both COCO and VOC datasets, DeFRCN outperforms the state of the art in all but one combination of novel classes and training images.

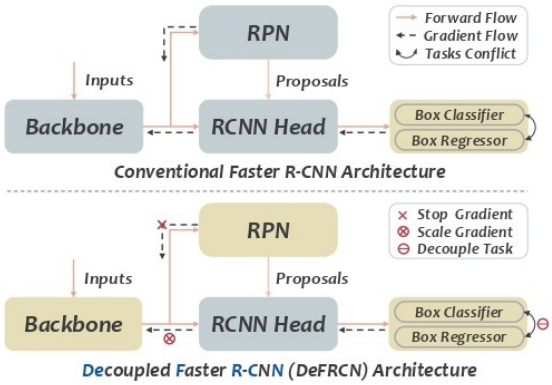

Figure 2.4: DeFRCN architecture. Image from Qiao et al. [9].

2.4.2 Meta-learning

Meta-learning is a learning paradigm that aims to learn how to learn. It is a form of transfer learning that learns to transfer knowledge from one task to another. In this section, we will go over the state of the art in few-shot object detection with meta-learning.

2.4.2.1 Meta R-CNN: Towards General Solver for Instance-level Low-shot Learning (Yan et al. [14])

Yan et al. [14] proposes a meta-learning approach to few-shot object detection. It extends Faster /Mask R-CNN[7] by introducing a Predictor-head Remodeling Network (PRN). PRN receives few-shot objects drawn from base and novel classes with their bounding boxes or masks, inferring class-attentive vectors corresponding to the classes that few-shot input objects belong to. These class-attentive vectors are then applied to take channel-wise attention on each RoI feature map, after which a binary detection outcome is predicted. The PRN architecture is shown in figure 2.5.

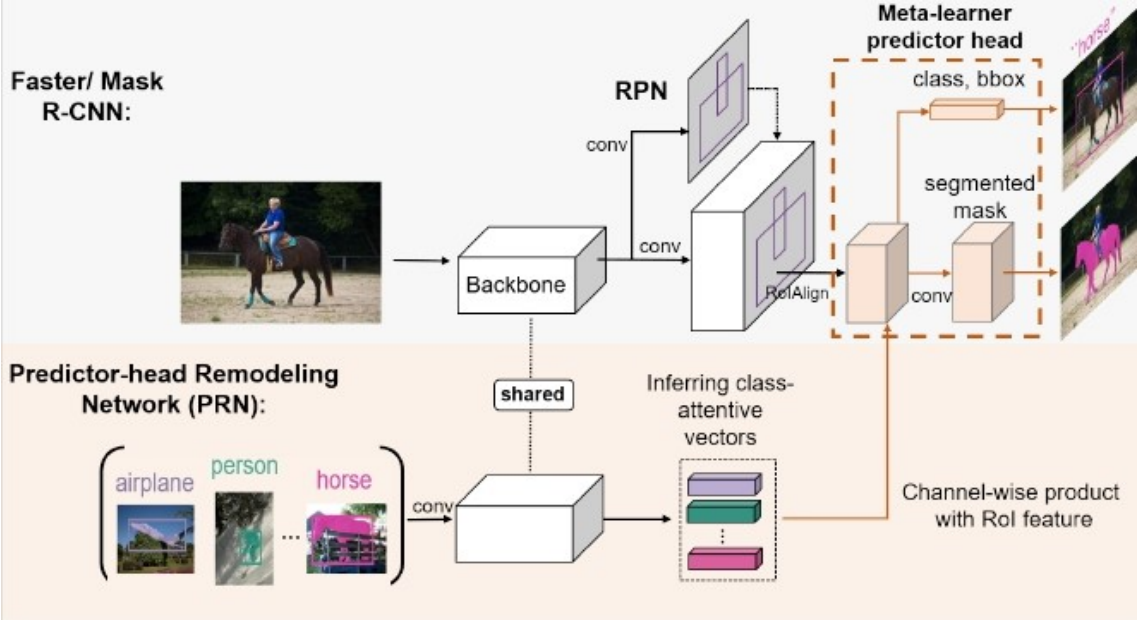

Figure 2.5: Meta R-CNN architecture with PRN. Image from Yan et al. [14].

2.4.2.2 Meta-DETR: Image-Level Few-Shot Detection with Inter-Class Correlation Exploitation (Zhang et al. [15])

Zhang et al. [15] proposes the first few shot object detector to work on the image level, shown in 2.6. It extends Deformable DETR with a Correlational Aggregation Network (CAN) to create Meta-DETR, shown in figure 2.7. The CAN architecture is shown in figure 2.8.

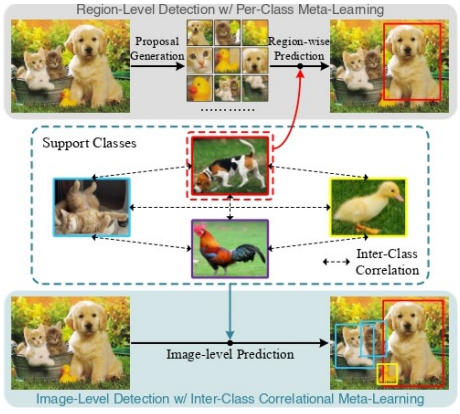

Figure 2.6: Image level detection. Image from Zhang et al. [15].

As shown in fig 2.8 the query and support features, extracted by ResNet-101, are first processed by a weight-shared multi-head attention module, encoding them into the same embedding space. Then the prototype for each support class is obtained by applying RoIAlign, followed by average pooling on the support features to get the support prototypes. Feature and encoding matching is then performed on the support prototypes and query features. The sum of the matching scores is

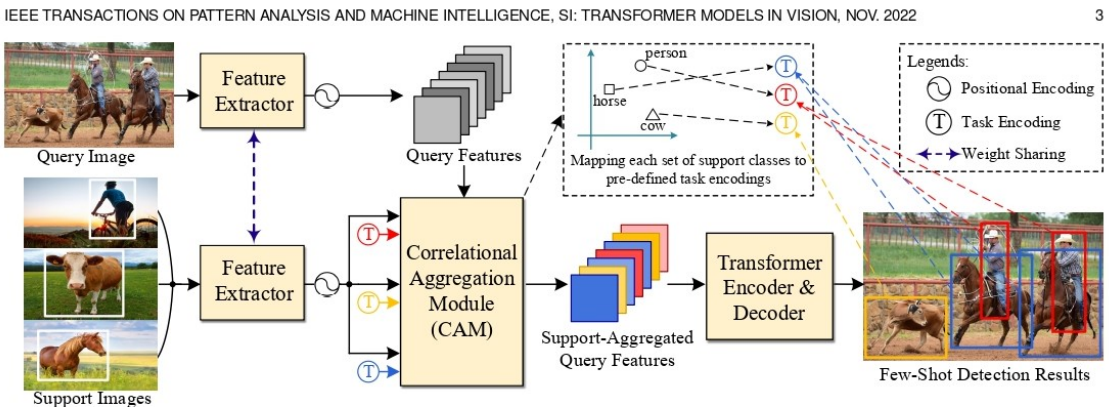

Figure 2.7: Meta-DETR architecture. Image from Zhang et al. [15].

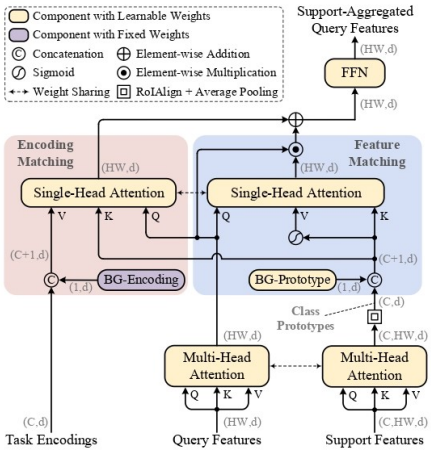

Figure 2.8: CAN architecture. Image from Zhang et al. [15].

then fed to a feed-forward network to obtain the final output.

2.5 Conclusion

In this chapter we have studied object counting, narrowing it down to few-shot object detection to count objects. We then went into more detail about the different ways to implement few-shot learning to detect objects. The sub-field of meta-learning shows the most promise for our problem as it is able to learn the difference between two alike novel classes better than transfer learning.

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] Hao Chen, Yali Wang, Guoyou Wang, and Yu Qiao. LSTD: A low-shot transfer detector for object detection. *CoRR*, abs/1803.01529, 2018. URL http://arxiv.org/abs/1803.01529.

[4] 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.

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

[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, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. *CoRR*, abs/1703.06870, 2017. URL http://arxiv.org/abs/1703.06870.

[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] Limeng Qiao, Yuxuan Zhao, Zhiyuan Li, Xi Qiu, Jianan Wu, and Chi Zhang. Defrcn: Decoupled faster R-CNN for few-shot object detection. *CoRR*, abs/2108.09017, 2021. URL https://arxiv.org/abs/2108.09017.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] Xiaopeng Yan, Ziliang Chen, Anni Xu, Xiaoxi Wang, Xiaodan Liang, and Liang Lin. Meta R-CNN : Towards general solver for instance-level low-shot learning. *CoRR*, abs/1909.13032, 2019. URL http://arxiv.org/abs/1909.13032.

[15] Gongjie Zhang, Zhipeng Luo, Kaiwen Cui, Shijian Lu, and Eric P. Xing. Meta-detr: Image-level few-shot detection with inter-class correlation exploitation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–12, 2022. doi: 10.1109/TPAMI.2022.3195735.

[16] 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.

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