KU LEUVEN

FACULTY OF ENGINEERING TECHNOLOGY

DE NAYER (SINT-KATELIJNE-WAVER) CAMPUS

# Counting shells

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

**Tijs VAN KAMPEN**

Promotor: Prof. Dr. Ir. Toon Goedemé

Co-Promotor: Ing. Tanguy Ophoff

Master thesis submitted to obtain the degree of Master of Science in the engineering technology: E-ICT Software Engineer - option ICT

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

**Summary** iii

**Abstract** iv

**Contents** vi

**List of figures** vii

**1 Introduction** 1

**2 Literature Review** 3

2.1 Crowd Counting . . . 3

2.2 Object detection . . . 4

2.2.1 R-CNN [7] . . . 4

2.2.2 SPPNet [9] . . . 4

2.2.3 Fast R-CNN [6] . . . 5

2.2.4 Faster R-CNN [18] . . . 6

2.2.5 Vision Transformers [4] . . . 7

2.2.6 Transformers [21] . . . 7

2.3 Few-shot object detection . . . 8

2.3.1 Method . . . 8

2.3.2 Data . . . 9

2.4 Metrics . . . 10

2.5 State of the art . . . 11

2.6 Conclusion . . . 14

**3 Implementation** 15

3.1 Dataset . . . 15

3.1.1 COCO . . . 15

3.1.2 Shells . . . . . 15
3.2 Baseline . . . . . 17
3.2.1 Model . . . . . 17
3.2.2 Inference . . . . . 18
3.2.3 Processing . . . . . 18
3.2.4 Results . . . . . 19

# List of Figures

1.1 The current process of collecting data.
Marked blue is the volunteer's actions, and marked red are the parts that involve experts. . . . 1
1.2 The new process of collecting data. . . . 2

2.1 RCNN architecture. Image from Girshick et al. [7]. . . . 4
2.2 SPPNet architecture. Image from He et al. [9]. . . . 5
2.3 Conventional CNN vs SPPNet. Image from He et al. [9]. . . . 5
2.4 Fast R-CNN architecture. Image from Girshick [6]. . . . 6
2.5 Faster R-CNN architecture. Image from Ren et al. [18]. . . . 6
2.6 ViT architecture. Image from Dosovitskiy et al. [4]. . . . 7
2.7 Attention layer. Image from Vaswani et al. [21]. . . . 8
2.8 Dual branch meta learning. Image from Köhler et al. [10]. . . . 9
2.9 IoU and class match to find the type of detection. . . . 10
2.10 Original ViT to OWL-ViT. Image from Minderer et al. [15]. . . . 12
2.11 imTED architecture. Image from Liu et al. [13]. . . . 12
2.12 Detailed view of the HAM. Image from Park and Lee [17]. . . . 13
2.13 Meta-contrastive learning. Image from Park and Lee [17]. . . . 13

3.1 Overview of the annotations per image in the shell dataset. . . . 16
3.2 Precision and recall for different IoU thresholds. . . . 19
3.3 Max precision and recall for different confidence thresholds. . . . 20
3.4 Mean and max precision and recall for different IoU thresholds. . . . 20
3.5 Precision and recall for filtering excessively large boxes. . . . 21

# 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.

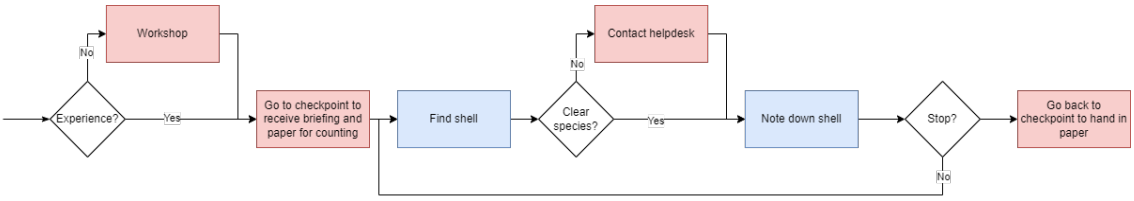



**Figure 1.1:** The current process of collecting data.
Marked blue is the volunteer's actions, and marked red are the parts that involve experts.

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[11], 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.

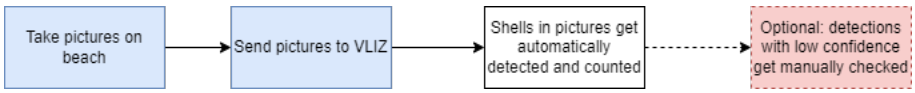



**Figure 1.2:** The new process of collecting data.

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[25] and COWC[16]. 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[8]. 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 [7]

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[20], 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.

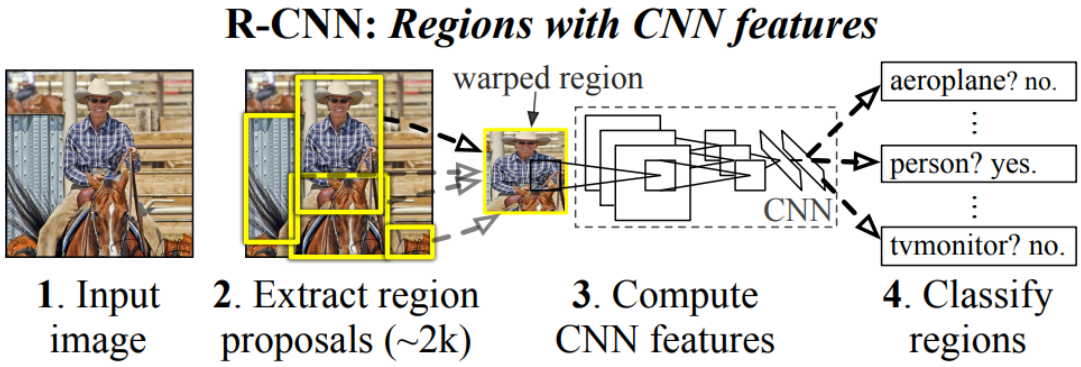



**Figure 2.1:** RCNN architecture. Image from Girshick et al. [7].

### 2.2.2 SPPNet [9]

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.

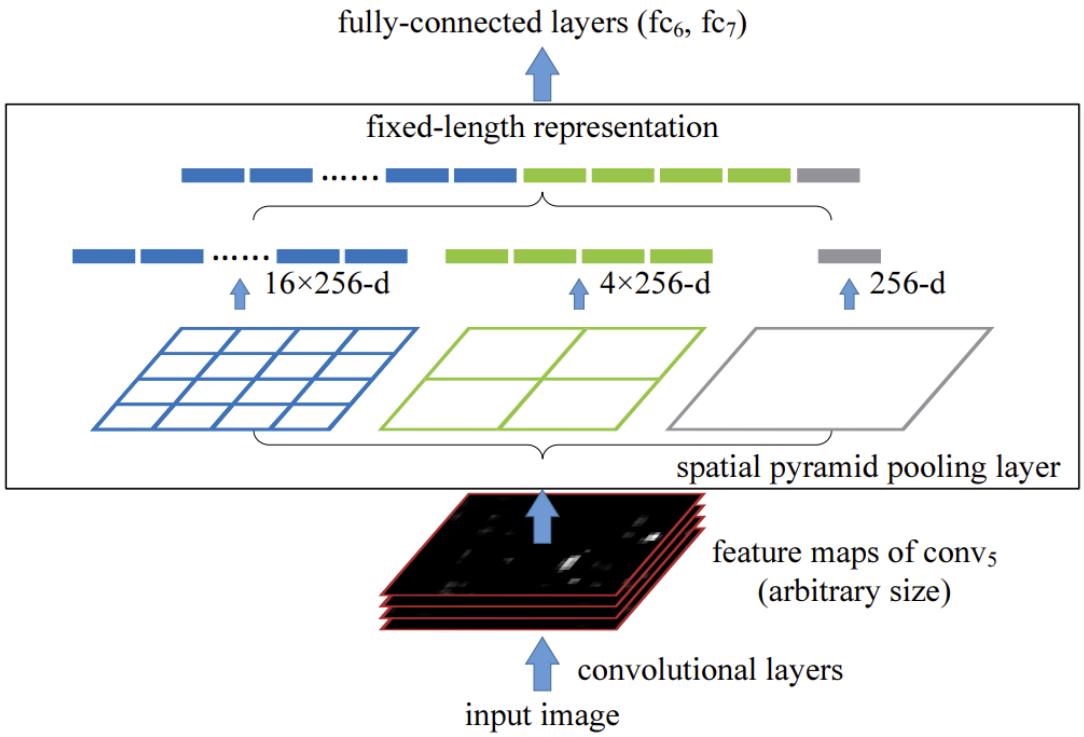



**Figure 2.2:** SPPNet architecture. Image from He et al. [9].

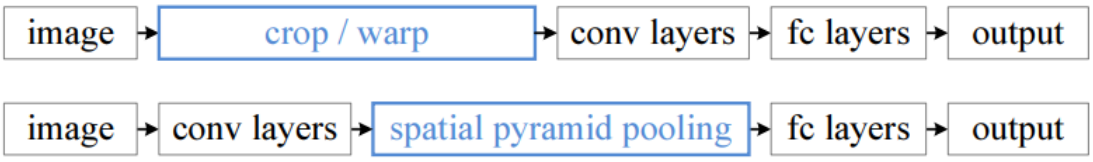



**Figure 2.3:** Conventional CNN vs SPPNet. Image from He et al. [9].

### 2.2.3 Fast R-CNN [6]

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.

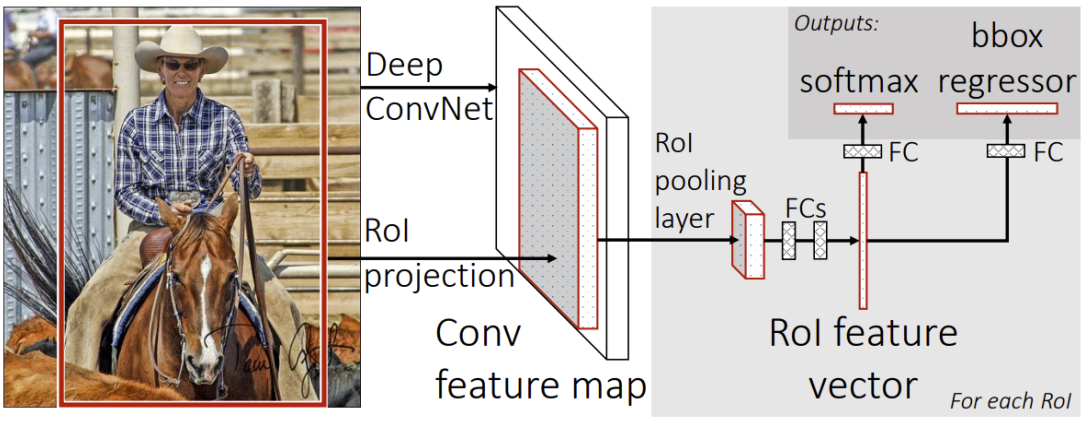



**Figure 2.4:** Fast R-CNN architecture. Image from Girshick [6].

### 2.2.4 Faster R-CNN [18]

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.

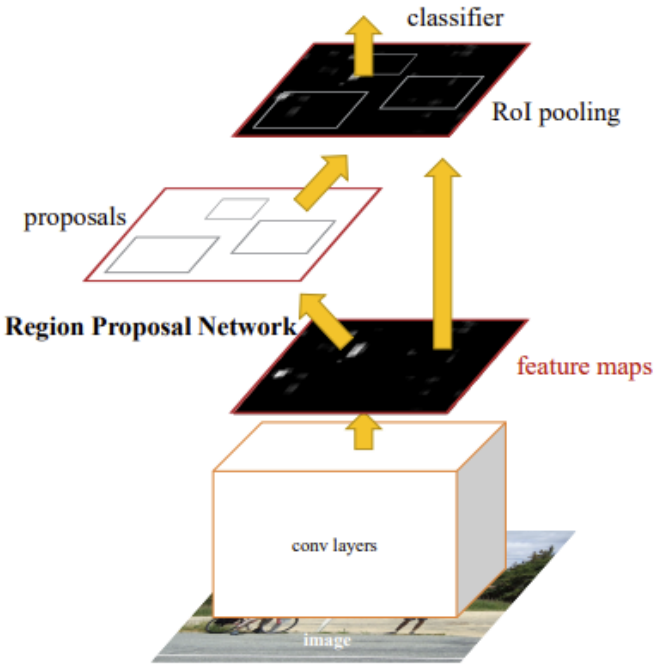



**Figure 2.5:** Faster R-CNN architecture. Image from Ren et al. [18].

### 2.2.5 Vision Transformers [4]

In recent years, the availability of more data has led to the development of architectures that were not previously possible. One example is the transformer architecture [21], originally designed for natural language processing (NLP) tasks, but now also used in computer vision by Dosovitskiy et al. [4] who call their approach Vision Transformers (ViT). ViT applies the standard transformers architecture to images with minimal changes and therefore shares most characteristics with transformers. Transformers performs best when trained on large amounts of data. ViT is pre-trained on either the ImageNet-21k dataset or the JFT-300M dataset and then fine-tuned on various benchmarks, where it exceeds the CNN-based state-of-the-art. The ViT architecture is illustrated in Figure 2.6. The model starts by dividing the image into a fixed number of patches, which are linearly projected into a 1D vector. To account for the position of each patch, a position embedding is added to the input.

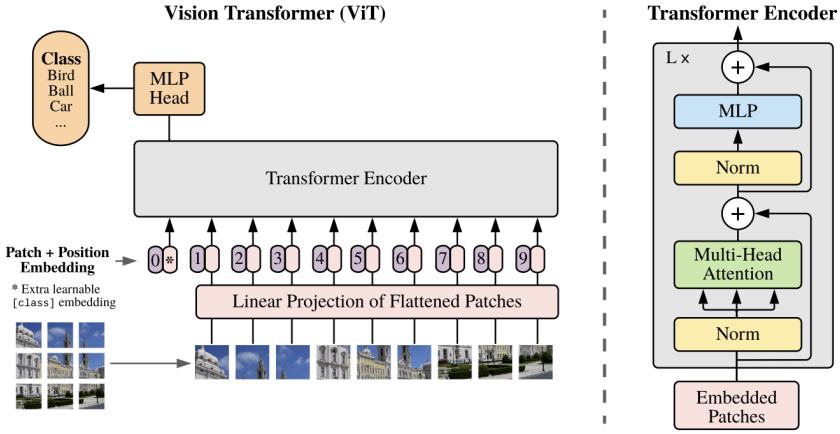



**Figure 2.6:** ViT architecture. Image from Dosovitskiy et al. [4].

### 2.2.6 Transformers [21]

As ViT uses transformers, it is important to understand how they work. The transformers architecture was introduced by Vaswani et al. [21] in Attention is all you need. It uses a traditional encoder-decoder structure but foregoes all convolutional layers for attention layers and fully connected layers. The encoder consists of two sub-layers for each layer, a multi-head self-attention layer and a fully connected feed-forward network. Each sub-layer has a residual connection and a layer normalization layer. The decoder is similar to the encoder but has an additional multi-head attention layer that takes the output of the encoder as input.

The attention layer is the most important part of the transformer architecture. It is a function that takes a set of queries, keys, and values and outputs a weighted sum of the values. The weights are calculated by taking the dot product of the query and key vectors and then applying a softmax function. The output of the attention layer is then the weighted sum of the values. The multi-head attention layer is a set of attention layers that are run in parallel, each with its own set of queries,

keys, and values. The output of each attention layer is concatenated and then linearly projected to the expected dimension. The attention layer is illustrated in Figure 2.7.

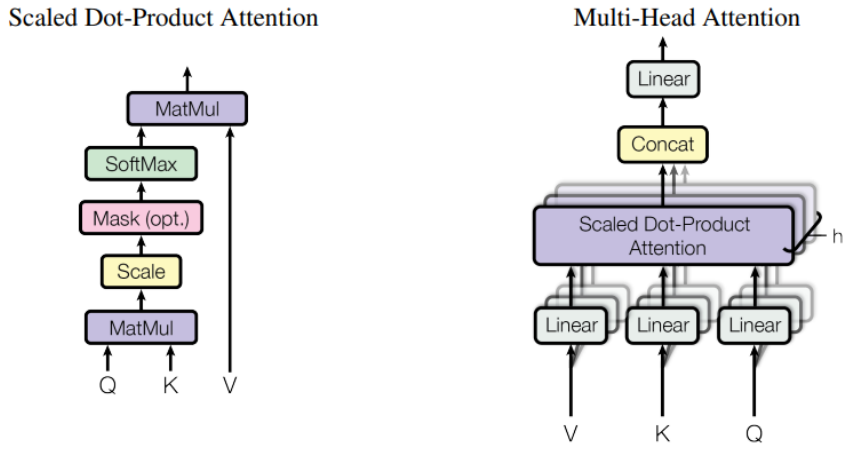



**Figure 2.7:** Attention layer. Image from Vaswani et al. [21].

## 2.3 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.

### 2.3.1 Method

Two different methodologies can be applied to few-shot learning, transfer learning and meta-learning. Each of these has its 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. [24, 22, 5, 19, 23].

### 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[10], displayed in figure 2.8. 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.

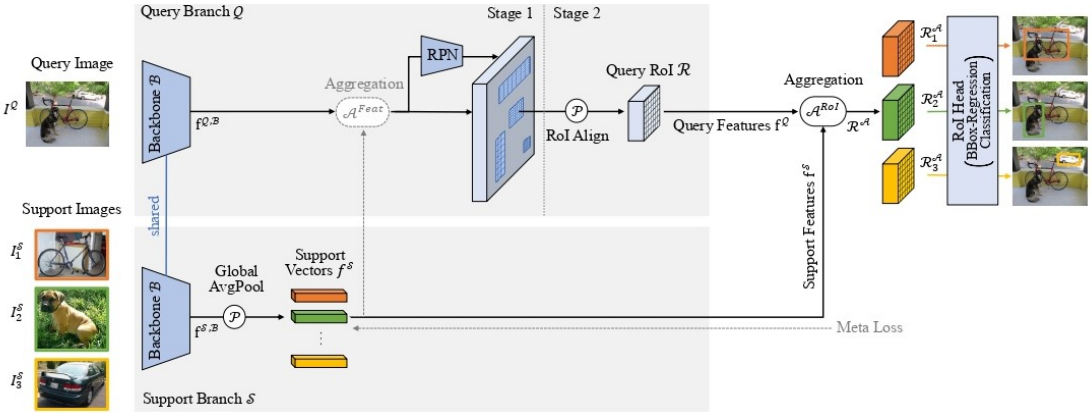



**Figure 2.8:** Dual branch meta learning. Image from Köhler et al. [10].

## 2.3.2 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 [14]. However, recently OWL-ViT [15] 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 2.5.

## 2.4 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.

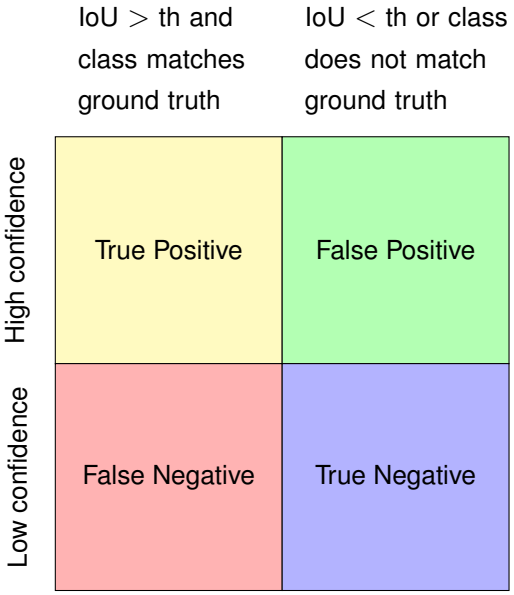



**Figure 2.9:** IoU and class match to find the type of detection.

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

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.

## 2.5 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. [15])

Minderer et al. [15] introduce a new method, Vision Transformer for Open-World Localization, or OWL-ViT. They modify ViT to make it work for object detection instead of classification. They modify ViT after pre-training by removing the token pooling layer, which downsamples the output embeddings to optimize computation, as that doesn't work for open-vocabulary detection. It is 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.

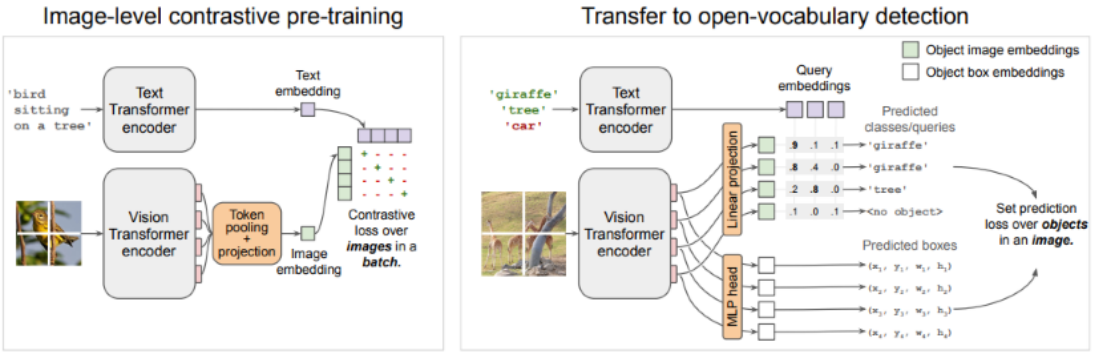



**Figure 2.10:** Original ViT to OWL-ViT. Image from Minderer et al. [15].

### Integrally Migrating Pre-trained Transformer Encoder-decoders for Visual Object Detection. (Liu et al. [13])

Liu et al. [13] 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 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. The final architecture is like faster RCNN, but with the encoder as the backbone and the decoder as the detector head. The number of parameters that are randomly initialized is reduced by 81.3%.

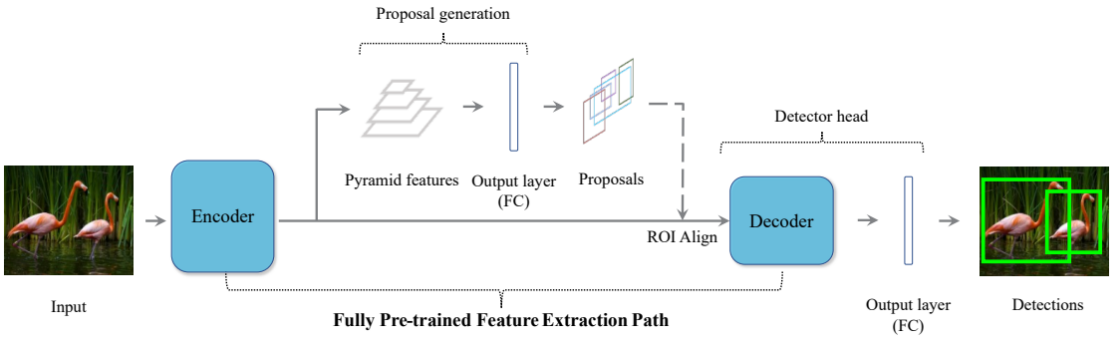



**Figure 2.11:** imTED architecture. Image from Liu et al. [13].

### Hierarchical Attention Network for Few-Shot Object Detection via Meta-Contrastive Learning. (Park and Lee [17])

Park and Lee [17] expand upon Faster R-CNN [18] 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 2.12. 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.

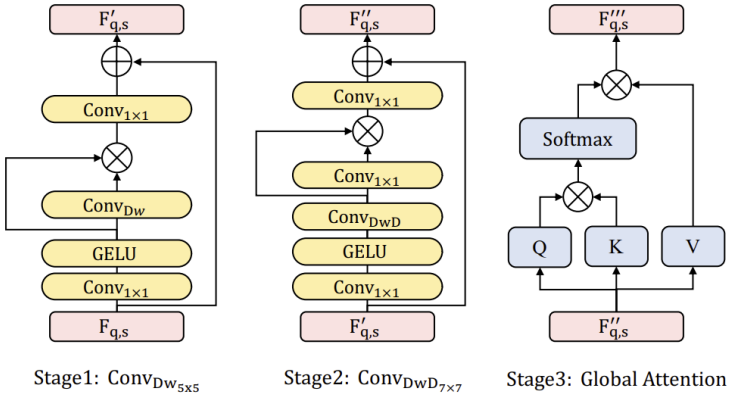



**Figure 2.12:** Detailed view of the HAM. Image from Park and Lee [17].

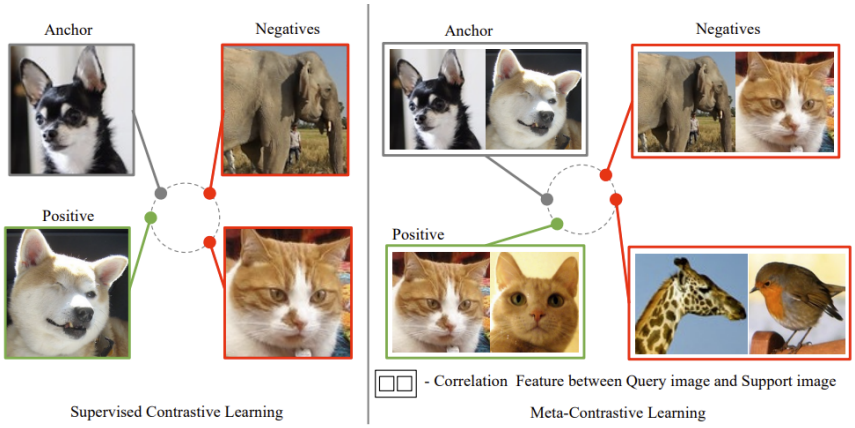



**Figure 2.13:** Meta-contrastive learning. Image from Park and Lee [17].

## 2.6 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/diffuculty |
| | 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] | Contrastive-learning suited to divide metric space of our alike shells |

*Results on AP50

**Results on AP COCO

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, an attention-based backbone. Both OWL-ViT as hANMCL can be used to initially establish a baseline as they do not require retraining. For our implementation we will start with finetuning hANMCL, if we have the time and hardware to do so we will also finetune OWL-ViT and imTED to compare the results.

# 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, both the datasets used to pre-train the models and the few-shot shell dataset we are introducing ourselves.

### 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. [12]

### 3.1.2 Shells

The shell dataset is a new dataset we are introducing ourselves. The images were taken with a cellphone camera on the Belgian coast. Each image has a size of 6144x8192px. Due to limited resources, the dataset is quite small, containing only 96 annotated images and 206 unannotated images. The images are annotated with bounding boxes of 8 types of shells, with 614 annotations in total. An overview of the annotations per class can be found in table 3.1. The images are mostly sparse, with 1 to 3 shells per image. Some denser images are also present, however. A graph of the annotations per image in the dataset can be found in figure 3.1. Some examples of the images in the dataset can be found in table 3.2.

| Shell type | Number of annotations |
|---|---|
| Baltic tellin | 69 |
| Cockle | 89 |
| Thick trough shell | 91 |
| Mussel | 268 |
| Banded wedge shell | 32 |
| Elliptical trough shell | 18 |
| Cut trough shell | 36 |
| Oyster | 9 |

**Table 3.1** Overview of the annotations per class in the shell dataset.

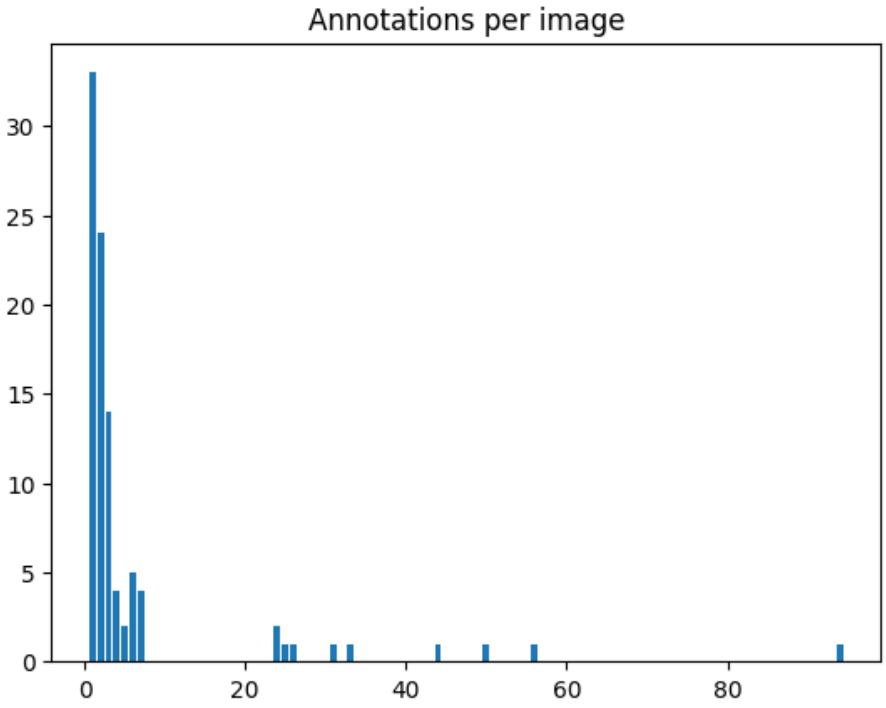



**Figure 3.1:** Overview of the annotations per image in the shell dataset.

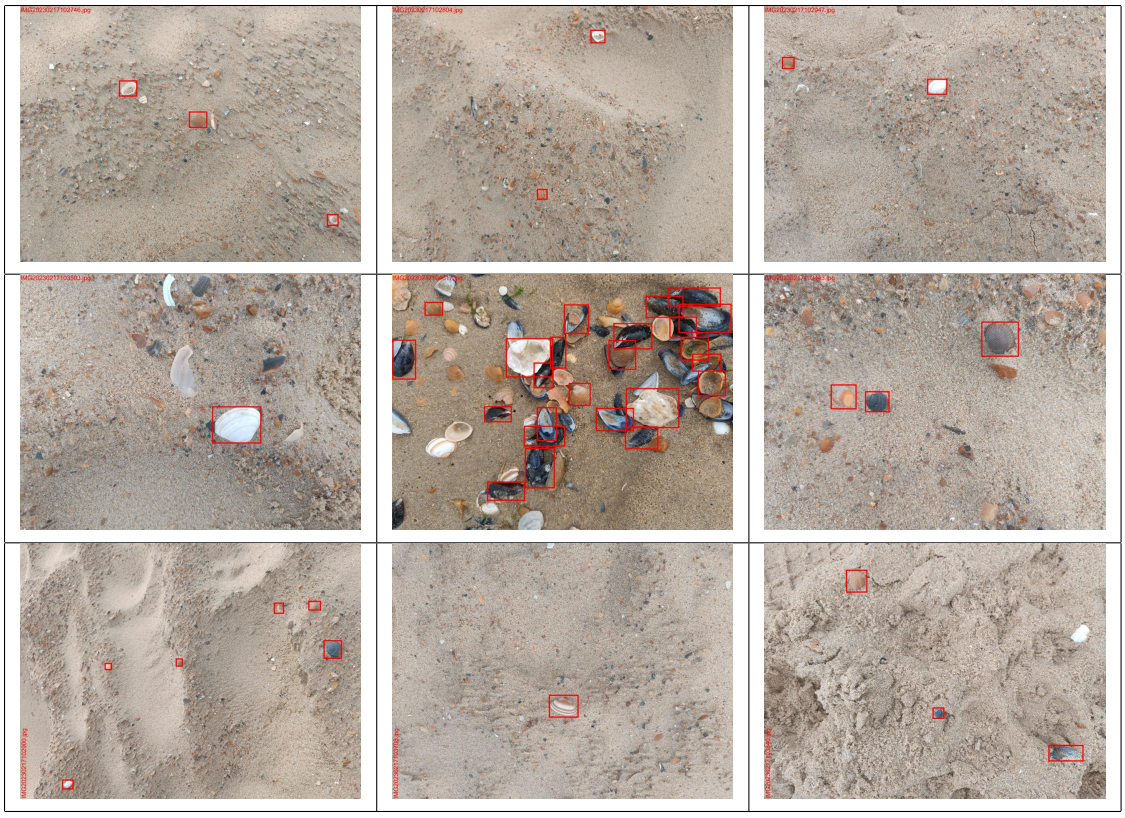


**Table 3.2** Examples of images in the shell dataset.

## 3.2 Baseline

In this section, we establish a baseline for the shell dataset. For this, we will study the performance of OWL-ViT with image-based one-shot detection.

### 3.2.1 Model

For the baseline, we will use the OWL-ViT model. A short description is made in section 2.5. At inference time embeddings are extracted from the image using the ViT encoder. These object image embeddings are then compared to the query (text or image) embeddings as shown in figure 2.10 to obtain a similarity score. The model exists in different flavors, with different backbones and different patch sizes. Hardware limitations limit us to the smallest ViT-B/32 model.

The model is implemented using the Scenic library [3]. It is also included in the transformers library, which offers an abstraction layer for running the model. Running the model is done by loading the processor and model, using the processor on the query images and test image to obtain the inputs for the model and then running the model.

### 3.2.2 Inference

Setting up for inference follows the following steps:

1. Load the model and processor.
    - The model and processor are readily available from the transformers library.
2. Load the images and annotations.
    - The images are Pascal VOC annotated, they are loaded into a class that offers the needed functionality.
3. Extract a query image for each class from the annotated images.
    - The query images are extracted from the first image that contains an annotation of that class.
    - Optionally the images can be shuffled to obtain a different query image for each class.
    - The image from which the query image was taken is then removed from the test set.
4. (Optional) Remove the background from the query images.
    - The background is removed using the rembg library, which uses a neural network (u2net) to remove the background.

Inference is then done by doing image-guided detection with all the query images on each of the test images sequentially due to hardware limitations.

### 3.2.3 Processing

To process the results, we can do the following operations [with these parameters]:

- NMS [True, False]
    - IoU threshold [0.1, 0.2, ..., 0.9]
- Apply confidence threshold [0.01, 0.02, ..., 0.99]
- Filter excessively large boxes [True, False]
    - We found that the model produces many excessively large boxes, which are filtered out if this parameter is set.

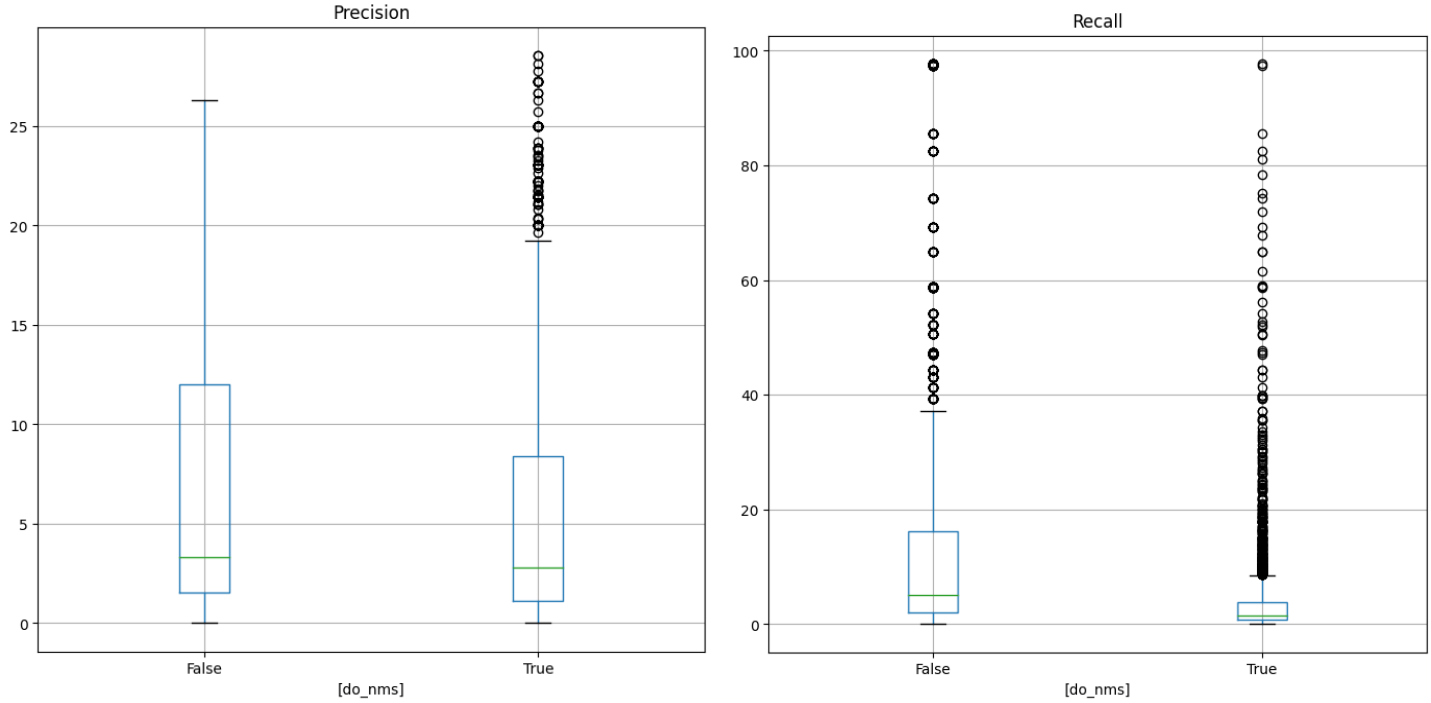
To find the optimal parameters, we create a grid of all possible combinations of these parameters. For each of the 4444 combinations, we process the model outputs. This returns the number of correct detections, correct bounding boxes, wrong detections and missed detections for each combination.

To evaluate the results we use the metrics described in section 2.4. Correct detections are the true positives, wrong detections and correct bounding boxes are the false positives and missed detections are the false negatives. We could calculate the metrics for getting the bounding box correct by taking the correct detections and the correct bounding boxes as true positives and only the wrong detections as false positives, but that is not relevant to our use case.

By calculating the metrics for each combination of parameters, we can discover the impact of each parameter on the performance of the model.

As seen in figure 3.2, enabling NMS will in most cases remove more true positives than false

positives, and thus decrease the performance of the model. Some outliers do however show that NMS can improve the precision of the model in some cases.




**(a)** Precision

**(b)** Recall

**Figure 3.2:** Precision and recall for different IoU thresholds.

As seen in figure 3.3, raising the confidence threshold will predicably reduce the recall while increasing the precision. The recall drops sharply, however, dropping to 20% at a confidence threshold of 0.2. Though increasing the confidence threshold does increase the precision as it eliminates more false than true positives, the recall drops so sharply that the overall performance of the model is reduced beyond the realm of usefulness.

As seen in figure 3.4, the IoU threshold has a large impact on the recall of the model. As fewer bounding boxes are eliminated, the recall increases with a higher threshold. The precision takes little to impact from varying the IoU threshold, making a good case for disabling NMS altogether.

Looking at figure 3.5, we see that filtering excessively large boxes improves precision by a large margin, with little to no impact on the recall. The model produces many excessively large boxes, which are filtered out if this parameter is set. The caveat is that this is technically cheating, as in the current implementation the solution is being used for this filtering. This could be improved upon by using a different approach to find appropriate measurements for filtering excessively large boxes.

### 3.2.4 Results

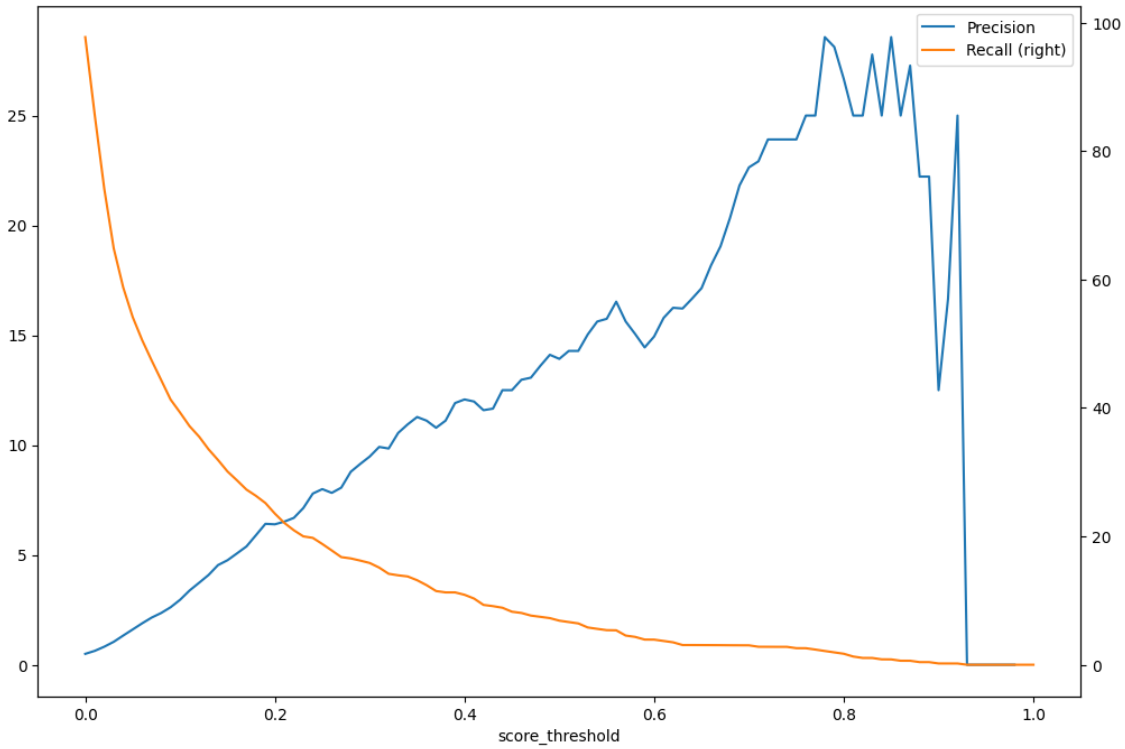
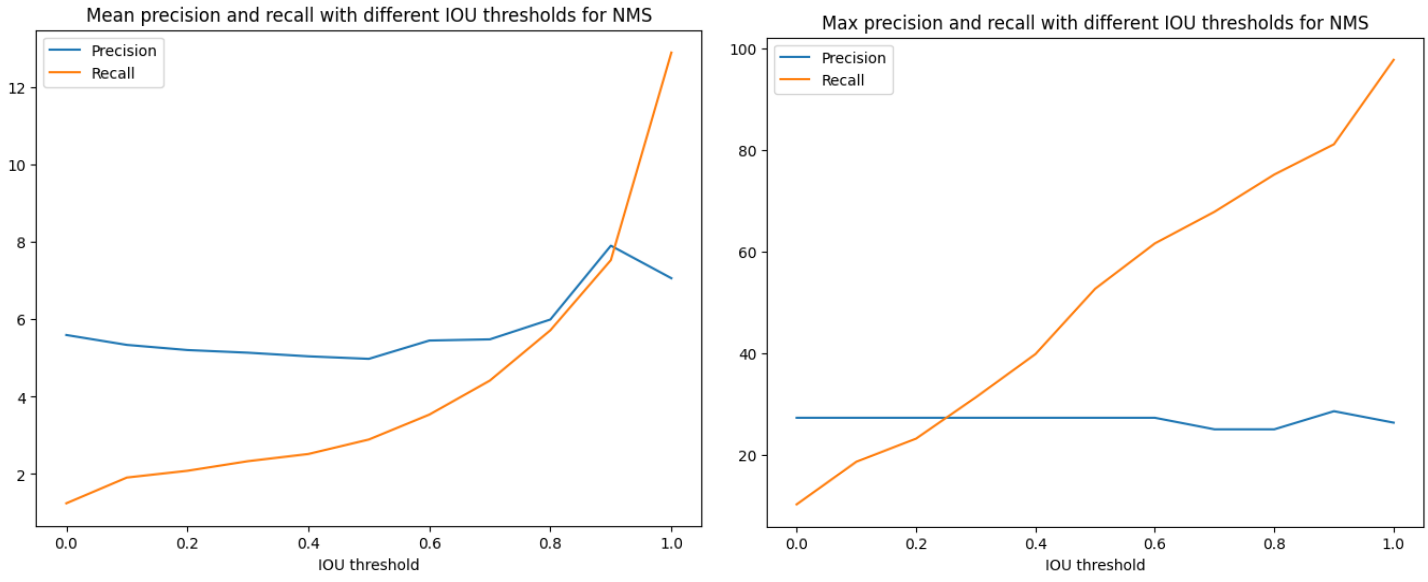


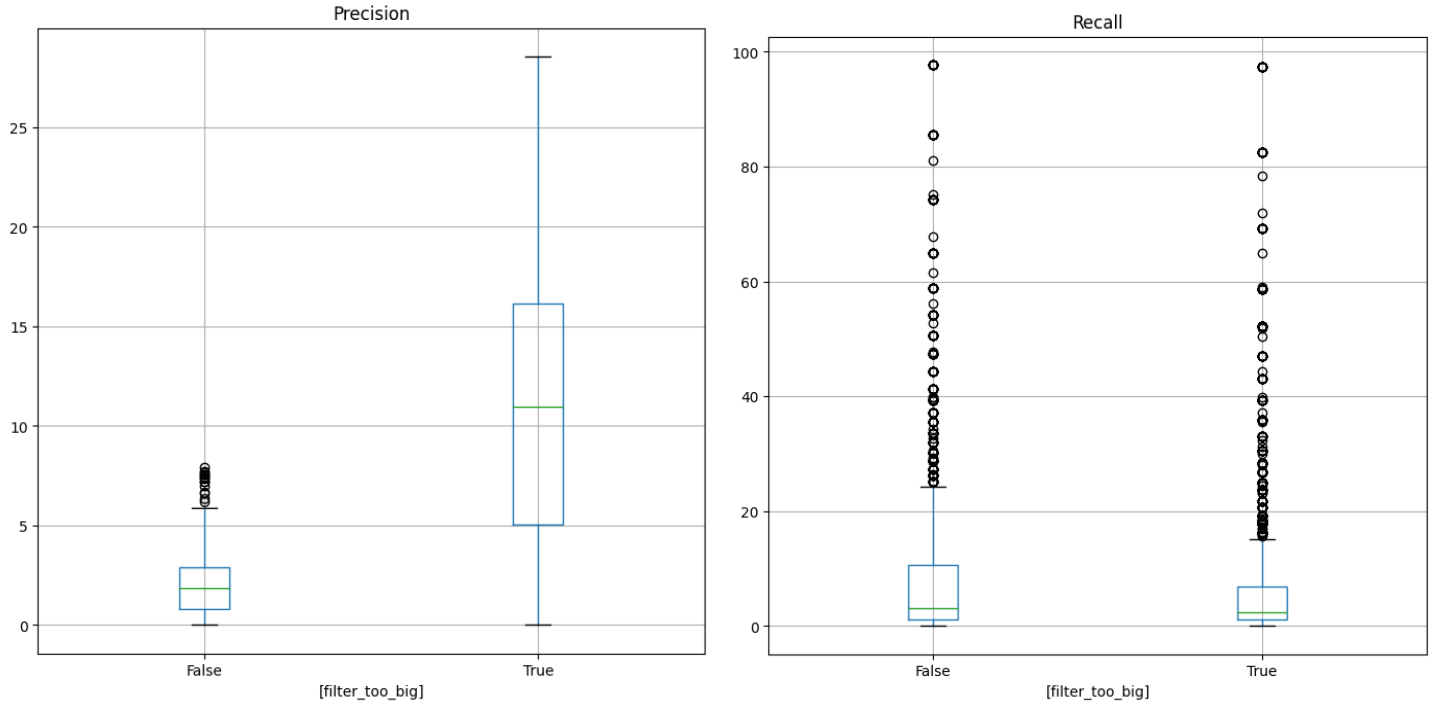

**Figure 3.3:** Max precision and recall for different confidence thresholds.




**(a)** Mean precision

**(b)** Mean recall

**Figure 3.4:** Mean and max precision and recall for different IoU thresholds.




**(a)** Precision

**(b)** Recall

**Figure 3.5:** Precision and recall for filtering excessively large boxes.

# 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] Mostafa Dehghani, Alexey Gritsenko, Anurag Arnab, Matthias Minderer, and Yi Tay. Scenic: A jax library for computer vision research and beyond. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 21393–21398, 2022.

[4] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *CoRR*, abs/2010.11929, 2020. URL https://arxiv.org/abs/2010.11929.

[5] 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.

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

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

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

[18] 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.

[19] 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.

[20] 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.

[21] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017. URL http://arxiv.org/abs/1706.03762.

[22] 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.

[23] 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.

[24] 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.

[25] 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
DE NAYER (SINT-KATELIJNE-WAVER) CAMPUS
Jan De Nayerlaan 5
2860 SINT-KATELIJNE-WAVER, België
tel. + 32 16 30 10 30
fet.denayer@kuleuven.be
www.fet.kuleuven.be