# Few Shot Learning for Object Detection

Alaa Arbi
Technical University of Munich
alaa.arbi@tum.de

## Abstract

*Few-shot Object Detection is a new research field that has gained a lot of attention because humans are capable of learning discriminative features of new objects from a few samples, while deep learning frameworks still require a large-scale dataset with abundant training samples to achieve reasonable results. In this report, we select two frameworks, DeFRCN and MetaDETR, and provide an in-depth understanding of their working principles. We split the object detection task into localization and classification tasks, present the challenges they face when only a few training samples are provided, and explain how each of the selected approaches proposes to solve it. Experiments on two benchmarks show that the selected frameworks achieved state-of-the-art results compared to previous works.*

## 1. Introduction

Deep neural networks have recently achieved state-of-the-art performance on various computer vision tasks, such as image classification [15], semantic segmentation [13], and object detection [21]. However, the performance of these methods is dependent on the use of large-scale datasets with high variability training data, making their application to real-world scenarios with limited annotated training samples challenging [6]. In some cases, collecting new annotated data is costly or even impossible, such as in the medical domain [6]. Conversely, humans can learn and generalize from just a few examples [6].

As a result, few-shot object detection (FSOD) has emerged as a new and promising research area. FSOD aims to detect (localize and classify) novel objects with only a few annotated instances in the target domain, leveraging the wealth of object detection benchmarks. According to [6], FSOD techniques can be categorized into meta-learning and transfer-learning based approaches, depending on the training paradigm.

In this report, we focus on two state-of-the-art FSOD methods, DeFRCN [14] and MetaDETR [22]. Despite using different network architectures and training paradigms, we highlight their common challenges and compare their proposed solutions.

## 2. Problem definition

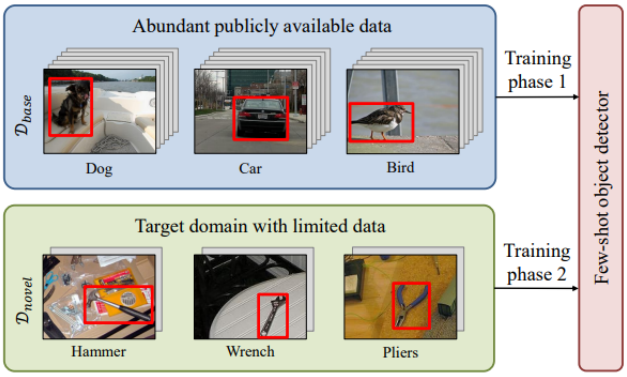


Figure 1. Illustration of the FSOD problem. In the first training phase the detector is trained on a large dataset then finetuned on the target data-scarce novel dataset. Taken from [6].

In the context of FSOD, a dataset $\mathcal{D} = \mathcal{D}_{base} \cup \mathcal{D}_{novel}$ is divided into two datasets: $\mathcal{D}_{base}$, which has abundant annotated instances, and $\mathcal{D}_{novel}$, which has K annotated instances for each of its N categories, hence the term "N-way K-shot object detection". Note that the base categories $\mathcal{C}_{base}$ in $\mathcal{D}_{base}$ are not overlapping with the novel categories $\mathcal{C}_{novel}$ in $\mathcal{D}_{novel}$, meaning $\mathcal{C}_{novel} \cap \mathcal{C}_{base} = \varnothing$. Each dataset is composed of tuples in the form $(I_i, \mathbf{y_i})$, where $I_i$ is an image containing M objects and $\mathbf{y_i} = (c_{i1}, b_{i1}), .., (c_{iM}, b_{iM})$ is a vector containing the labels for the M objects present in the image, including the category $c_{ij}$ and the bounding box annotations $b_{ij}$. Also, multiple instances of objects from the same category can be present in the same image.

The goal of FSOD is to train a model based on $\mathcal{D}$ so that it can correctly localize and classify objects in a new set $\mathcal{D}_{query}$. There are two cases:

- $\mathcal{C}_{query} = \mathcal{C}_{novel}$: in which the model does not have to maintain its performance on the base categories.

- $\mathcal{C}_{query} = \mathcal{C}_{novel} \cup \mathcal{C}_{base}$: in which the model must accurately detect objects of novel categories while still maintaining its performance on the base categories. This special case is referred to as Generalized-FSOD.

A visual representation of the FSOD problem definition can be seen in Figure 1.

## 3. Related Work

### 3.1. General Object Detection

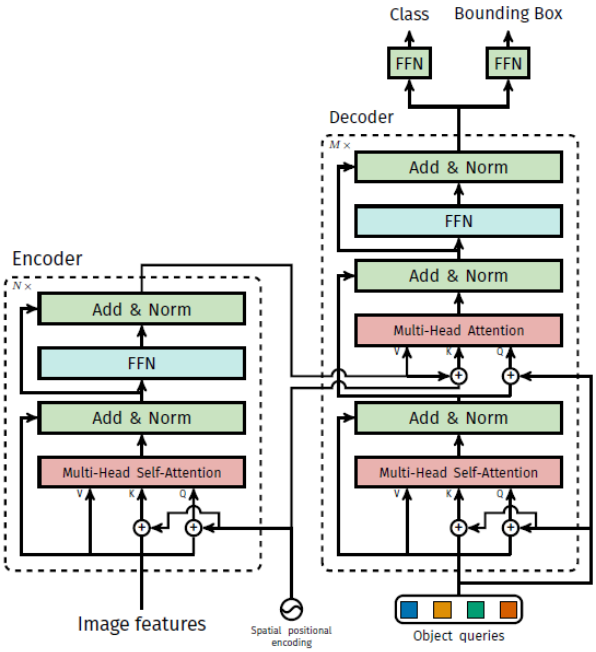


Figure 2. The transformer encoder-decoder architecture in DETR. Taken from [1].

Deep learning-based approaches for object detection can be split into three categories:

1. One-stage proposal-free methods: They use dense bounding boxes to generate predictions and apply post-processing, e.g., non-maximum suppression, to prune redundant predictions [21]. These methods are usually faster in inference time, yet not as accurate as other methods [14]. This approach includes SSD [10], YOLO [16], RetinaNet [8], SqueezeDet [19].

2. Two-stage proposal-based methods: They first predict regions of interest (ROIs) in the first stage and next classify them in the second one. This category includes Faster R-CNN [17], which is the foundation for many FSOD approaches such as DeFRCN [14]. It takes the input image, passes it to a shared backbone, then a Region Proposal Network is used to find ROIs that are pooled to a fixed size using RoIAling and fed to the RCNN head to classify the object present in the ROI and adjust the bounding box parameters. The Faster R-CNN [17] is optimized as a multi-task optimization problem using the loss function:

$$\mathcal{L}_{total} = (\mathcal{L}^{cls}_{rpn} + \mathcal{L}^{reg}_{rpn}) + (\mathcal{L}^{cls}_{rcnn} + \mathcal{L}^{reg}_{rcnn}) \quad (1)$$

3. Other approaches: like the DETR [1] that is an image level detector trained end-to-end doesn't explicitly use region proposals before classifying them. DETR [1] extracts features from the input image then passes them to a transformer-based encoder-decoder to output N object queries that are then classified and their bounding boxes are regressed as illustrated in figure 2.

All these approaches however, share the assumption that a large dataset of annotated data is available, which may cause issues in a few shot detection problem [14].

### 3.2. Few-Shot Object Detection

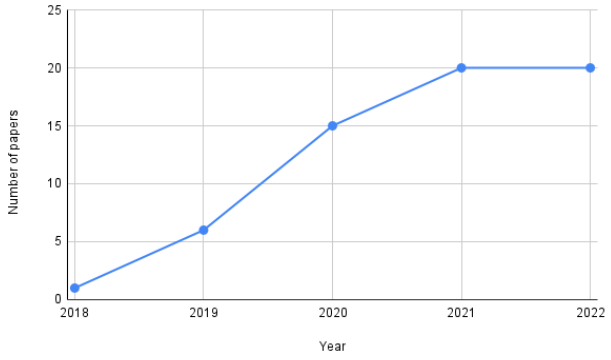


Figure 3. Number of papers used in the survey [6] per year of publication. It is clear that few shot object detection gained lot of interest in the recent years.

As previous detectors expect large amounts of annotated data, few-shot object detection has started to gain more and more interest, especially in the last years, as can be seen in figure 3. The vast majority of few shot detectors are build on the the Faster R-CNN [17] architecture e.g. [5, 7, 11, 12, 14, 18, 20]. TFA [18] uses a simple approach by just fine-tuning the last layer of the model and freezing all other parts. FSDetView [20] extracts features from the query image and from support class images, then applies a complex aggregation operation that combines subtraction with channel-wise multiplication and concatenation to guide the network. TIP [7] tackles the problem from a data-centric point of view. Intuitively applying data augmentation should help the model generalize better, in TIP [7] however, it was noticed that naively augmenting support images

reduced the model's performance. This is due to the guidance vectors of augmented support images being too different from the guidence vectors of the original ones although they should be transformation invariant. TIP [7] then used an additional transformed guidance consistency loss based on the $l^2$ norm to constrain support vectors of augmented classes to be close and thus representative for the class. DCNet [5] use a self-designed mechanism similar to cross attention in order to aggregate query features with support features, and after retrieving region proposals from the RPN they use different scales for the pooling operation in the RoIAlign block. Large resolutions are intended to focus on small objects, while smaller ones are for big-sized objects. IFC [12] uses an interactive self-attention module to capture the differentiating characteristics of scarce new categories and a special feature aggregation technique with the intention of reducing duplicate data and boosting feature stability and sensitivity for both novel and basic categories. In FCT [4] the ResNet backbone of Faster R-CNN [17] is replaced by a Pyramid Vision Transformer, and feature aggregation between query and support images is performed at multiple levels, namely at three stages in the backbone and one more time in the detection using a novel attention-based module called the fully cross-transformer. MemFRCN [11] pointed out the problem of catastrophic forgetting in few-shot detectors, i.e. the accuracy on base classes falls back as the detector is fine-tuned to detect novel ones. To circumvent this problem, MemFRCN [11] replaced the detection head of Faster R-CNN with a special one that is built of a memory block that stores the representative vector of each class and computes the similarity scores between them and region proposals, a fully connected classifier to output classification scores for the proposals, and a fusion block that would merge them with the similarity scores peviously predicted by the memory block.

## 4. Training Schemes

Following [6], the different approaches in FSOD can be split according to the training scheme into two categories, namely meta-learning and transfer-learning -based approaches. In both schemes, the training is split into two stages: base training, in which the initial model $\mathcal{M}_{init}$ is trained on the base dataset $\mathcal{D}_{base}$ to get the base model $\mathcal{M}_{base}$, followed by a few-shot fine-tuning stage, in which the base model $\mathcal{M}_{base}$ is trained on the fine-tuning data scarce dataset $\mathcal{D}_{finetune}$ resulting in the fine-tuned model $\mathcal{M}_{finetune}$ [6]. Note that if the model is supposed to keep its performances in the base dataset $\mathcal{D}_{base}$ then $\mathcal{D}_{finetune}$ is a balanced dataset containing both novel and base categories, else $\mathcal{D}_{base} = \mathcal{D}_{finetune}$ [6]. Thus, the problem setting can be summarized as follows:

$$\mathcal{M}_{init} \xrightarrow{\mathcal{D}_{base}} \mathcal{M}_{base} \xrightarrow{\mathcal{D}_{finetune}} \mathcal{M}_{finetune} \quad (2)$$

with $\xrightarrow{\mathcal{D}}$ indicating model training using dataset $\mathcal{D}$.

In the following, we highlight the specifics of each procedure:

- Meta-Learning: The training is split into E episodes [6]. In each episode, the model is trained using K instances of N categories sampled randomly from the training set [6]. This procedure is intended to force the model to learn to discriminate different categories using few examples [6]. Meta-DETR [22] is trained using this scheme and fine-tuned using a balanced set of novel and base categories.

- Transfer-Learning: After the base training stage, the weights of some parts of the model are frozen, and the other weights are then fine-tuned in the second stage [6]. One drawback of this training technique is that since almost all weights of the network are frozen the model may be incapable to cope with the severe shift in data distribution when finetuned on the novel set and thus fail to generalize [14]. DeFRCN [14] is trained this way and finetuned once using $\mathcal{D}_{novel}$ only and once using a balanced set of novel and base categories.

## 5. Identified Problems, and Solutions

The task of object detection is composed of two subtasks: localizing the object and classifying it. The overall performance of the detector is determined by the accuracy of both of them, as a correctly localized but falsely classified object is considered a false detection, and poor localization (e.g., pointing to the background as an object) affects the classification power of the model. In the following, the two sub-tasks are considered separately, and it is described how DeFRCN [14] and MetaDETR [22] suggested to improve their accuracy.

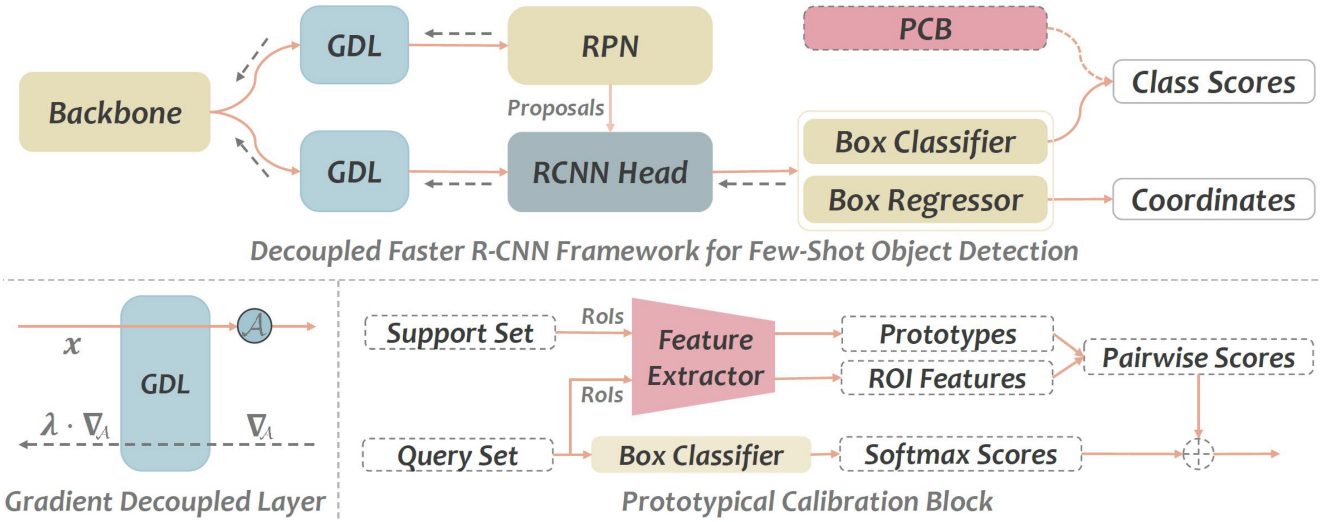


Figure 4. The general architecture of the Decoupled Faster R-CNN with an illustration of the Gradient Decoupled layer and the Prototypical Calibration Block

### 5.1. Localization

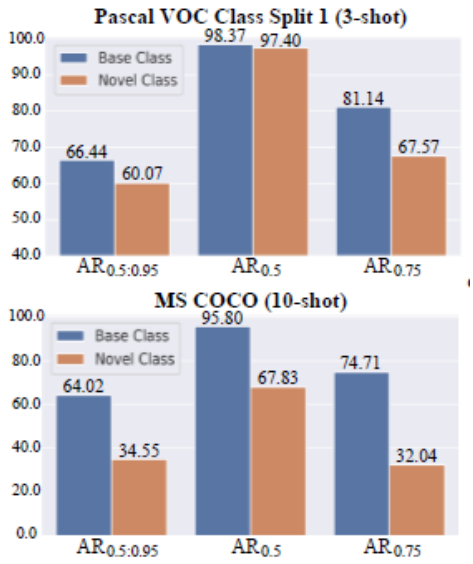


Figure 5. The localization quality for novel classes (The measure is Average Recall on the top 1000 region proposals) is obviously poorer than that of base classes due to the few number of training samples. Taken from [22].

According to [22], there is a glaring difference in the quality of area proposals for base and novel classes, as seen in figure 5. Therefore, region-based techniques like those built upon the Faster R-CNN architecture are inherently bound by the inaccurate localization suggestions for novel classes that are due to the extremely limited number of training samples [22]. This is impeding the detector's performance on novel classes.

DeFRCN [14] revisited the original architecture of Faster R-CNN [17] and pointed to two major conflicts that could be hindering the model's performance, especially in the FSOD context: The network can be seen as three interdependent and interacting modules (namely the shared backbone, the RPN, and the RCNN) that are jointly optimized in a multi-task learning scheme following the loss equation 1. The RPN tries to localize objects independently of their classes, the RCNN classifies them and optimizes the bounding box parameters [14]. While the RPN and the regression head of the RCNN require translation covariant input features, the classification head of the RCNN expects translation invariant input features [14]. And while the RPN expects class-agnostic input features, the RCNN requires class-specific features [14]. However, both of them are using the output features of the shared backbone that learns to output generic sub-optimal features [14]. Moreover, during base training, the RPN considers objects corresponding to novel classes as background, and thus it might influence the backbone to ignore them and over-fit to base classes, which makes it harder for the fine-tuned model to generalize to the novel dataset [14]. These conflicts arise from the joint optimization of the model's components that influence each other during optimization [14]. Therefore, DeFRCN [14] proposed to use two Gradient Decoupled Layers (GDL) that are inserted into the architecture of Faster R-CNN in order to decouple its three modules (the backbone, the RPN, and the RCNN head) as can be seen in figure 4. One GDL is inserted between the backbone and the RPN while the other one is located between the backbone and the RCNN. To easily improve feature represen-

tations and accomplish forward-decoupling, GDL uses an affine transformation layer $\mathcal{A}$, parameterized by learnable channel-wise weights $w$ and bias $b$, during forward propagation [14]. The transformation layer $\mathcal{A}$ maps therefore the shared backbone feature to new different input features that are specific and optimal to the RPN and the RCNN which can be mathematically expressed as follows [14]:

$$In_{rpn} = \mathcal{A}_{rpn}(Out_b),\ In_{rcnn} = \mathcal{A}_{rcnn}(Out_b.) \quad (3)$$

During the backwards propagation, GDL sends the gradient from the preceding layer to the succeeding layer after multiplying it by a constant $\lambda$ of a value between [0, 1] [14].Thus the new update rule for the backbone parameters is expressed as follows [14]:

$$\theta_b \longleftarrow \theta_b - \gamma(\lambda_{rpn}\frac{\partial \mathcal{L}_{rpn}}{\partial \theta_b} + \lambda_{rcnn}\frac{\partial \mathcal{L}_{rcnn}}{\partial \theta_b}) \quad (4)$$

The parameter update rules for the RPN and the RCNN are not changed as the GDL is placed before them [14]. Experiments with different lambda values ([0, 0.25, 0.5, 0.75, 1] in base training and [0, 0.001, 0.01, 0.1, 1] in fine-tuning) were conducted, and the best results were achieved when $\lambda_{RPN} = 0$ in both base and fine-tuning and $\lambda_{RCNN} = 0.75$ in base training, $\lambda_{RCNN} = 0.01$ in fine-tuning [14]. Meaning that the gradient from the RPN should be stopped and the gradient from the RCNN should be scaled to reduce its impact on the backbone [14].

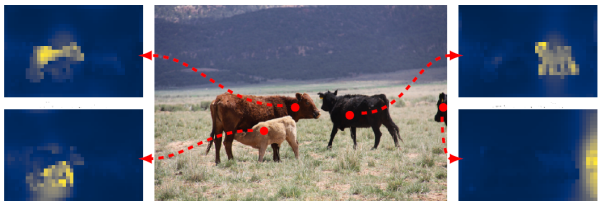

Figure 6. Visualization of encoder attention in DETR. Taken from [1].

On the other hand, MetaDETR [22] was based on the DETR [1] which is an image-level object detector. It avoids the limitation of faulty region suggestions in common few-shot detection systems since it does not rely on region proposals [22]. After extracting features (of shape HxWxd from the input image, DETR [1] uses a transformer encoder-decoder to predict $N$ object queries that are then classified and used to regress bounding boxes parameters as can be seen in figure 2. The encoder uses a multi-head self-attention to process image features and aggregate the relations of image areas to each other (e.g. the part of an image containing a cow's head should have a high attention coefficient with the part containing the same cow's body and a small coefficient with every other part, as can be seen in figure 6) [1]. As transformers process sequences and they are permutation-invariant, the input image features are flattened to the shape HWxd and positional encoding is added to them [1]. Next, the decoder takes in $N$ object queries that are initialized to zero and learnt [1]. These object queries are again processed using a self-attention mechanism and added to the queries input of the second attention head, whose keys input is the encoder output [1]. After visualizing the decoder attention in figure 7, one sees that the decoder is assigning a high attention coefficient to pixels corresponding to object extremities [1].

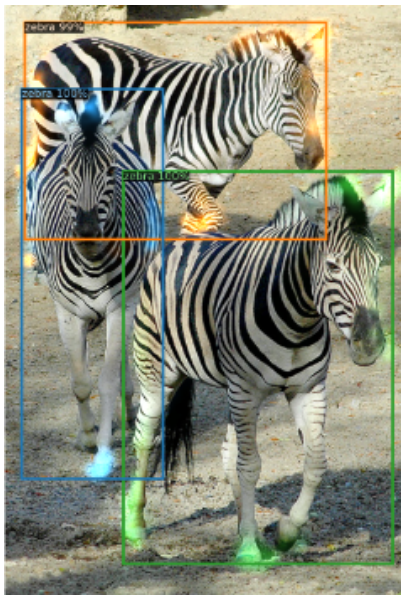

Figure 7. Visualization of the decoder attention in DETR. Taken from [1].

## 5.2. Classification

In DeFRCN [14] it was noticed that the classification scores of the few shot detectors are of low quality. It was conjectured that this is due the localization branch of the RCNN that might be forcing the backbone to learn translation covariant features which on the other hand degrades the performance of the classification branch [14]. Therefore, they proposed the Prototypical Calibration Block (PCB) in order to refine the scores in inference time [14]. The PCB first extracts some features from an image using a classifier that was pretrained on ImageNet [14]. Then, using the bounding box coordinates and RoIAlign layer, it extracts the respective features for each object in the image [14]. In the context of a M-way K-shot task, the PCB thus extracts MK feature vectors, K for each of the M categories [14]. The feature vectors are then averagely pooled per category to get M category prototypes that are stored in a prototype bank $\mathcal{P} = \{p_c\}_{c=1}^{M}$ [14]. In inference time the same process is applied to the input image (the used bounding box coordinates are the predicted ones) to get a feature vector $x_i$ for

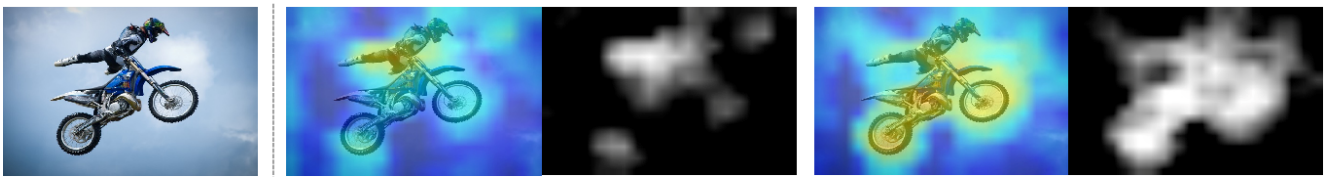


Figure 8. The visualization of PCB on a test image. The white pixels in the similarity map indicate a high similarity between the features corresponding to that area and the prototype of the class indicated by a ⋆. Taken from [14]

each object [14]. The cosine similarity score between $x_i$ and each class prototype $p_c$ is computed and used to refine the classification score as follows:

$$s_i^{refined} = \alpha s_i + (1 - \alpha) s_i^{cos} \tag{5}$$

with $\alpha$ being a weighting factor that is set to 0.5 [14]. To prove the efficacity of the PCB, the cosine similarity between the class prototypes and the feature map extracted from a test image is computed and visualized in figure 8 in which it can be seen that the similarity is high in areas that correspond to the same class [14].

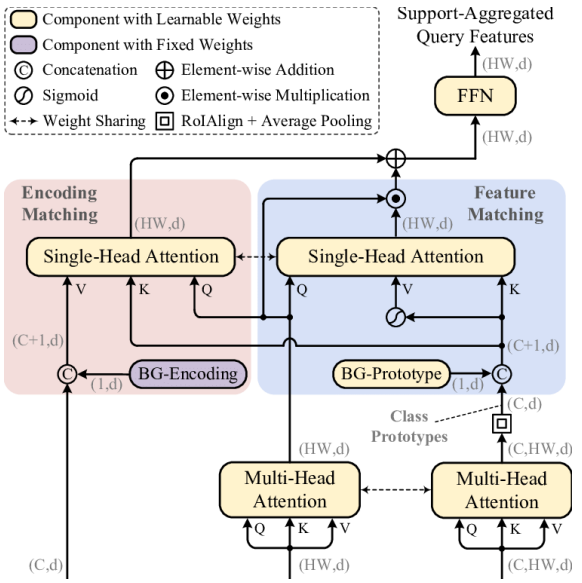


Figure 9. The Correlation Aggregation Module CAM presented in MetaDETR. Taken from [22].

In MetaDETR [22] it was noticed that items of classes with a high visible similarity, e.g. ("cows","horses"), ("dogs","cats") and ("motorbike","bicycle") are often missclassified due to ignorance of the high inter-class correlation. Therefore, they proposed the Correlation Aggregation Module (CAM) that uses attention mechanisms to simultaneously aggregate query features extracted from the input images together with support class prototypes leveraging inter-class correlation to reduce missclassification and generalize better to novel classes [22]. Features are extracted from the query image and the sampled support images using a shared feature extractor and then sent to the CAM in order to aggregate the query features with the correlation information [22]. The CAM illustrated in figure 9 first processes the query and support features using a shared Multi-Head-Attention [22]. Just like in the PCB in [14], RoIAlign and average pooling are applied to the resulting support features to get the class prototypes [22]. A background prototype (BG-Prototype) is then appended to the class prototypes [22]. This BG-Prototype is learned during training as there is no explicit representation of the class background [22]. Next CAM uses a weight-shared Single-Head-Attention mechanism to apply:

- Feature Matching: It is intended to discard features irrelevant to the support classes [22]. The $HW$ d-dimensional vectors of the query features are compared to the $C+1$ d-dimensional class prototypes using cosine similarity [22]. The matching coefficients give the attention matrix $A$ whose element $a_{ij}$ indicates the similarity between the i-th query feature and the j-th class prototype: the higher it is, the more similar they are. Then the attention matrix is applied to the value vectors $V = \sigma(Class - Prototypes)$ to get a filtered feature map $Q_F$ that contains only features that are highly correlated with the support classes [22]. $Q_F$ is then aggregated again with the original query features using the Hadamard product [22]. The process is mathematically expressed as follows:

$$Q_{FM} = Q \odot A \cdot \sigma(S) \tag{6}$$

with $A = Attn(Q, S)$ being the attention matrix [22].

- Encoding Matching: For each class as well as the background class, a special encoding vector following the spatial encoding scheme of the transformers is presented [22]. The same single-head-attention mechanism used in feature matching is again applied here with the only difference that the values $V$ are now set to the pre-defined class-encodings [22]. While the class prototypes change according to the sampled sup-

port images, the class encodings are constant.

$$Q_{FE} = S \odot A \cdot T \tag{7}$$

The resulting features are finally summed together and processed using a feed-forward-network to output the correlation enhanced features that are now the input to the transformer encoder-decoder model [22]. Alongside with using bipartite matching loss like in [1], in MetaDETR [22] a cosine-similarity cross-entropy loss is applied to the CAM-generated class prototype to force them to be discriminative. Note that during inference, no images for support classes are repeatedly sampled. Instead, the class prototypes are learned once and used for detection on every query image to guarantee efficient inference.

# 6. Datasets, Evaluation Protocols and Results

## 6.1. Metrics

Accuracy, precision, recall, F1 score are all metrics that can be used for classification. For localization, one can use IoU, for examlpe. In object detection classification and localization are simultaneously considered thus special metrics need to be defined. Considering a single class, one can compute the following:

1. Compute the Intersection over Union: which is the ratio of the intersection area to the union area between the ground truth bound boxes and the predicted bounding boxes

2. For a specific IoU threshold $\alpha$ compute the number of True Positives TP, number of False Positives FP, number of False Negatives FN which are defined as (note that multiple detections of the same object in an image were considered false detections, e.g. 5 detections of a single object counted as 1 correct detection and 4 false detections):

   - TP: a detection for which $IoU \geqslant \alpha$
   - FP: a detection for which $IoU < \alpha$
   - FN: a missed ground truth detection

3. Compute Precision: which is the ratio of the correct detections to all detections

$$Precision = \frac{TP}{TP + FP} \tag{8}$$

4. Compute Recall: which is the ration of the correct detections to ground truth

$$Recall = \frac{TP}{\#GroundTruth} \tag{9}$$

5. Compute the Average Precision for class $c$ at IoU threshold $\alpha$ $AP^c_\alpha$ which is the tradeoff score between Precision and Recall and can be seen as the area under the Precision-Recall-Curve (which takes into consideration the model's confidence score).

Once the class-specific Average Precisions are computed, the mean Average Precision (mAP) score can thus be computed by averaging over all class scores:

$$mAP_\alpha = \sum_{c=1}^{C} AP^c_\alpha \tag{10}$$

In PASCAL VOC the $mAP_\alpha$ is simply noted as $AP_\alpha$ and in the COCO dataset $mAP$ is the average of the $mAP_\alpha$ scores over different IoU thresholds $\alpha$ values.

## 6.2. PASCAL VOC Dataset

The PASCAL VOC Dataset [2] is a benchmark for object detection, containing objects from 20 different categories. To use it for the problem of few-shot object detection three sets are created where each set it split into $\mathcal{D}_{base}$ and $\mathcal{D}_{novel}$ with $|\mathcal{C}_{base}| = 15$ and $|\mathcal{C}_{novel}| = 5$ [6]. K, the number of shots for the novel categories, is set to 1, 2, 3, 5, and 10. The evaluation metric is the mean average precision at an intersection over union (IoU) threshold of 0.5($AP_{50}$). VOC12+07 train/val sets are used to create the training set, while the VOC07 test set is kept for testing.

While categories in DeFRCN [14] are randomly divided into base classes and novel classes, the splits in MetaDetr [22] are the following:

1. $\mathcal{C}_{novel} = bird, bus, cow, motorbike, sofa$
2. $\mathcal{C}_{novel} = aeroplane, bottle, cow, horse, sofa$
3. $\mathcal{C}_{novel} = boat, cat, motorbike, sheep, sofa$

And the specific K-shots that were used for training were randomly sampled and not specified. The reported results are averaged over multiple iterations (10 for MetaDETR [22] and not specified for DeFRCN [14])

The results of different approaches on this dataset are presented in table 1. Even though the class splits might not be the same, as sometimes they are done randomly and not explicitly indicated, it is assumed that the results are comparable. We see from table 1 that DeFRCN [14] in both settings FSOD and G-FSOD (the latter being marked by the letter g) dominates the best first and second best results with a margin of +0.1 (Set 3, K=1 compared to MetaDETR [22]) to +9.1 (Set 3, K=2, compared to MetaDETR [22]). Note that in the FSOD setting, an increase in the number of shots does not necessarily lead to an increase in performance, as performance dropped from 64.1 to 60.8 when increasing K from 5 to 10 in novel set 1 and similar in novel set 2. In [14]

it was conjectured that this is due to the quality of the randomly sampled training examples which can have a negative effect if they are of low quality.

### 6.3. Microsoft COCO Dataset

The Microsoft COCO Dataset [9] is a more challenging benchmark for object detection with 80 categories. In the context of few-shot object detection, the 20 categories of PASCAL VOC are considered the novel categories, and the remaining 60 categories are the base ones [6]. The number of shots K is set to 1, 3, 5, 10, and 30. The evaluation metric is the mean of 10 mean average precisions $mAP$ for COCO $AP_{50:95}$ computed at different IoU thresholds [0.5 to 0.95 with a step size of 0.05]. While in MetaDETR [22] train set 2017 is used for training, and val set 2017 is used for evaluation, in DeFRCN [14] only 5k images from the val set 2017 are used for testing, and the rest are used for training. The reported results are averaged over multiple iterations (5 for MetaDETR [22] and not specified for DeFRCN [14]).

The results of different approaches on this dataset are showen in table 2. MetaDETR [22] and DeFRCN [14] (in both settings FSOD and G-FSOD ) consistently outperform other frameworks. In fact for a number of shots equal to 10 and 30, MetDETR [22] achieved an improvement of +2.3 and +2 respectively and DeFRCN [14] had showed a boost of +1.8 and +2.4 respectively. Note that compared to TFA [18] DeFRCN [14] has much more learnable parameters and still achieves better results as it manages to adapt to the data distribution of novel classes without over-fitting to base ones.

Because the Microsoft COCO dataset is more complex than the PASCAL VOC dataset in terms of classes, occlusions, and scale variants, we can safely say that the selected two approaches outperform all other works.

**Which one is better, DeFRCN or MetaDETR ?** Based on the results on COCO dataset, one could naively say that DeFRCN is outperforming MetaDETR for different numbers of shots K. However, note that MetaDETR was finetuned using the G-FSOD setting and it is outperforming DeFRCN [14] in the same setting for novel classes. As results for MetaDETR in G-FSOD for base classes were not reported like in DeFRCN one can not simply say that MetaDETR is better in this setting as it might be performing worse on the base classes. Regarding the FSOD training, the MetaDETR was not even trained in this setup, and thus no results were reported.

Table 2. Results on Microsoft COCO dataset. Colors RED/BLUE respectively indicate the best and second best results.

| | Number of Shots K | | | | |
|---|---|---|---|---|---|
| | 1 | 3 | 5 | 10 | 30 |
| MetaDETR [22] | 7.5 | 13.5 | 15.4 | 19.0 | 22.2 |
| DeFRCN [14] | 9.3 | 14.8 | 16.1 | 18.5 | 22.6 |
| DeFRCN g [14] | 4.8 | 10.7 | 13.6 | 16.8 | 21.2 |
| TFA [18] | 1.9 | 5.1 | 7.0 | 9.1 | 12.1 |
| FCT [4] | 5.1 | 9.8 | 12.0 | 15.3 | 20.2 |
| MemFRCN [11] | 5.2 | 8.3 | 10.8 | 14.0 | 17.5 |
| FSDetView [20] | - | - | - | 12.5 | 14.7 |
| IFC [12] | - | - | - | 16.7 | 17.9 |
| TIP [7] | - | - | - | 16.3 | 18.3 |
| DCNet [5] | - | - | - | 12.8 | 18.6 |

**Comparaison to rich annotated datasets**

In table 3 we compare the performance of DeFRCN trained with 30 shots on the COCO dataset to that of four other detectors that are using Faster R-CNN and DETR architectures. Even though DeFRCN did not outperform them, its accuracy is still fairly competitive (especially compared to the Res2Net [3] published in 2019) with regard to the limited number of training samples it used for finetuning. MetaDETR results for all the classes of COCO dataset were not reported and thus were not included here in the comparison.

### 6.4. Ablation Study

#### 6.4.1 Catastrophic forgetting in G-FSOD

As can be seen in the second and third rows of table 2, DeFRCN [14] consistently achieves better results on novel classes when trained in the FSOD setting. This is probably due to the fact that in a G-FSOD setting, the network must keep its performance on the 60 base classes, which makes it harder to optimize it to the novel ones without catastrophic forgetting happening. It could be suggested here to take inspiration from techniques presented in MemFRCN [11] to reach better results on novel classes without forgetting base ones by adding the scores of a memory classifier to the score refinement process.

#### 6.4.2 GDL in rich data annotation scenarios

As proposed in DeFRCN [14] it seems that the GDL prevents the Faster R-CNN network from being dominated by one of its three components, which results in a boost in the FSOD problem. This gave the authors the intuition to apply it to a data-rich detection problem, and they showed that GDL resulted in an improvement of +0.8, +1.5, +1.2 for the metrics $AP$, $AP_{50}$, $AP_{75}$ repectively when added to the original FRCN with a ResNet50 backbone that was trained

Table 1. Results on PASCAL VOC dataset. Colors RED/BLUE respectively indicate the best and second best results.

| | | MetaDETR [22] | DeFRCN [14] | DeFRCN g [14] | TFA [18] | FCT [4] | MemFRCN* [11] | FSDetView [20] | IFC [12] | TIP [7] | DCNet [5] |
|---|---|---|---|---|---|---|---|---|---|---|---|
| mean | | 45.8 | 51.9 | 49.4 | 34.7 | 46.4 | 35.2 | 36.7 | 45.4 | 38.5 | 39.2 |
| Novel Set 1 | 1 | 35.1 | 53.6 | 40.2 | 25.3 | 38.5 | 36.4 | 24.2 | 41.0 | 27.7 | 33.9 |
| | 2 | 49.0 | 57.5 | 53.6 | 36.4 | 49.6 | 37.4 | 35.3 | 47.9 | 36.5 | 37.4 |
| | 3 | 53.2 | 61.5 | 58.2 | 42.1 | 53.5 | 40.6 | 42.2 | 52.7 | 43.3 | 43.7 |
| | 5 | 57.4 | 64.1 | 63.6 | 47.9 | 59.8 | 45.5 | 49.1 | 55.0 | 50.2 | 51.1 |
| | 10 | 62.0 | 60.8 | 66.5 | 52.8 | 64.3 | 46.6 | 57.4 | 61.5 | 59.6 | 59.6 |
| Novel Set 2 | 1 | 27.9 | 30.1 | 29.5 | 18.3 | 25.9 | 18.0 | 21.6 | 28.6 | 22.7 | 23.2 |
| | 2 | 32.3 | 38.1 | 39.7 | 27.5 | 34.2 | 26.8 | 24.6 | 34.2 | 30.1 | 24.8 |
| | 3 | 38.4 | 47.0 | 43.4 | 30.9 | 40.1 | 32.1 | 31.9 | 36.8 | 33.8 | 30.6 |
| | 5 | 43.2 | 53.3 | 48.1 | 34.1 | 44.9 | 36.3 | 37.0 | 42.4 | 40.9 | 36.7 |
| | 10 | 51.8 | 47.9 | 52.8 | 39.5 | 47.4 | 32.4 | 45.7 | 49.4 | 46.9 | 46.6 |
| Novel Set 3 | 1 | 34.9 | 48.4 | 35.0 | 17.9 | 34.7 | 30.3 | 21.2 | 37.3 | 21.7 | 32.3 |
| | 2 | 41.8 | 50.9 | 38.3 | 27.2 | 43.9 | 32.3 | 30.3 | 43.5 | 30.6 | 34.9 |
| | 3 | 47.1 | 52.3 | 52.9 | 34.3 | 49.3 | 37.3 | 37.2 | 45.7 | 38.1 | 39.7 |
| | 5 | 54.1 | 54.9 | 57.7 | 40.8 | 53.1 | 37.8 | 43.8 | 50.7 | 44.5 | 42.6 |
| | 10 | 58.2 | 57.4 | 60.8 | 45.6 | 56.3 | 38.5 | 49.6 | 53.9 | 50.9 | 50.7 |

Table 3. Results of different detectors on the COCO 2017 val set (copied from [1]) for base and novel classes. In first block 4 detectors trained on rich annotated data are presented. We chose to compare them with DeFRCN trained with 30 shots to make results as comparible as possible

| | mAP |
|---|---|
| Faster R-CNN+ResNet50+FPN | 40.2 |
| Faster R-CNN+ResNet101+FPN | 42.0 |
| DETR+ResNet50 | 42.0 |
| DETR+ResNet101 | 43.5 |
| Res2Net+ResNet50 | 33.7 |
| DeFRCN+K=30 | 31.4 |

and evaluated on the COCO dataset. Similar improvements were seen when using ResNet101 as a backbone.

#### 6.4.3 PCB for other approaches

The PCB was shown to improve the performance of different settings of DeFRCN [14]. In fact, the authors trained the baseline FRCN architecture for FSOD and then equiped it with PCB, reporting an increase of +2.5 and +2.6 in the $mAP$ metric on COCO dataset for K equal 10 and 30, respectively. As it is a plug-and-play offline module, they used it in other approaches, e.g. TFA [18] and noticed an improvement ranging from +2.1 to +3.0 depending on the number of shots. Therefore, the PCB can be considered in any other approach to boosting performances.

#### 6.4.4 CAM to reduce inter-class confusion

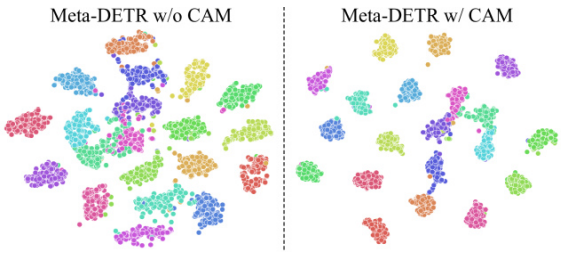


Figure 10. Visualization of object representations in feature space projected using t-sne when trained with and without CAM. Object from PASCAL VOC set 1 trained in 2-shot setting. Taken from [22].

CAM in MetaDETR [22] was proposed to leverage inter-class correlation to improve detection of similar looking classes. To prove its efficacity, object representations in feature space gained with and without CAM are plotted in figure 10 after being projected to 2D using t-SNE. It is clear that CAM better separates class objects from each other. Moreover, the confusion matrices for similar classes, namely "cow" vs. "horse" and "motorbike" vs. "bike" were observed. It was clear that CAM reduced the number of

missed detections and missclassifications, which proves the intuition that CAM reduces misclassification of similarly looking classes.

## 7. Conclusion

In this report, we presented an in-depth understanding of two few shot object detection frameworks that achieved state-of-the-art performances, namely DeFRCN [14] and MetaDETR [22]. The two approaches are trained using different training schemes, namely meta-learning and fine-tuning, and based on two different network architectures, namely the attention-based DETR [1] and the broadly used Faster R-CNN [17] architecture. We highlighted the innovative contributions that they presented and how they helped solve challenges that arise from the lack of abundant training data: DeFRCN [14] uses GDL layers to decouple the network components from each other, enabling the whole network parameters to be tuned without overfitting, and uses a PCB-based score refinement to improve classification and MetaDETR [22] uses the image level detector DETR to circumvent the problem of low quality region proposals and proposes a CAM module to aggregate image features with inter-class correlation information to improve classification power and help the model generalize to novel classes. Despite their simplicity, both reached better results on two common benchmark datasets: Pascal Voc [2] and Microsoft COCO [9].

## References


[1] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers, 2020. 2, 5, 7, 9, 10

[2] Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John M. Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88:303–338, 2010. 7, 10

[3] Shang-Hua Gao, Ming-Ming Cheng, Kai Zhao, Xin-Yu Zhang, Ming-Hsuan Yang, and Philip Torr. Res2net: A new multi-scale backbone architecture. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(2):652–662, feb 2021. 8

[4] Guangxing Han, Jiawei Ma, Shiyuan Huang, Long Chen, and Shih-Fu Chang. Few-shot object detection with fully cross-transformer, 2022. 3, 8, 9

[5] Hanzhe Hu, Shuai Bai, Aoxue Li, Jinshi Cui, and Liwei Wang. Dense relation distillation with context-aware aggregation for few-shot object detection. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10180–10189, 2021. 2, 3, 8, 9

[6] Mona Köhler, Markus Eisenbach, and Horst-Michael Gross. Few-shot object detection: A comprehensive survey, 2021. 1, 2, 3, 7, 8

[7] Aoxue Li and Zhenguo Li. Transformation invariant few-shot object detection. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 3094–3102. Computer Vision Foundation / IEEE, 2021. 2, 3, 8, 9

[8] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection, 2017. 2

[9] 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, 2014. 8, 10

[10] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: Single shot MultiBox detector. In *Computer Vision – ECCV 2016*, pages 21–37. Springer International Publishing, 2016. 2

[11] TongWei LU, ShiHai JIA, and Hao ZHANG. Memfrcn: Few shot object detection with memorable faster-rcnn. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E105.A(12):1626–1630, 2022. 2, 3, 8, 9

[12] Hongwei Ning Haipeng Liu Meng Wang. Object detection based on few-shot learning via instance-level feature correlation and aggregation, 2022. 2, 3, 8, 9

[13] Shervin Minaee, Yuri Boykov, Fatih Porikli, Antonio Plaza, Nasser Kehtarnavaz, and Demetri Terzopoulos. Image segmentation using deep learning: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(7):3523–3542, 2022. 1

[14] Limeng Qiao, Yuxuan Zhao, Zhiyuan Li, Xi Qiu, Jianan Wu, and Chi Zhang. Defrcn: Decoupled faster r-cnn for few-shot object detection, 2021. 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

[15] Waseem Rawat and Zenghui Wang. Deep convolutional neural networks for image classification: A comprehensive review. *Neural Computation*, 29(9):2352–2449, 2017. 1

[16] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection, 2015. 2

[17] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks, 2015. 2, 3, 4, 10

[18] Xin Wang, Thomas E. Huang, Trevor Darrell, Joseph E. Gonzalez, and Fisher Yu. Frustratingly simple few-shot object detection, 2020. 2, 8, 9

[19] Bichen Wu, Alvin Wan, Forrest Iandola, Peter H. Jin, and Kurt Keutzer. Squeezedet: Unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving, 2016. 2

[20] Yang Xiao, Vincent Lepetit, and Renaud Marlet. Few-shot object detection and viewpoint estimation for objects in the wild, 2020. 2, 8, 9

[21] Syed Sahil Abbas Zaidi, Mohammad Samar Ansari, Asra Aslam, Nadia Kanwal, Mamoona Asghar, and Brian Lee. A survey of modern deep learning based object detection models, 2021. 1, 2

[22] 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. 1, 3, 4, 5, 6, 7, 8, 9, 10