Few-Shot Object Detection: A Survey

SIMONE ANTONELLI, Sapienza Università degli Studi di Roma, IT
DANILO AVOLA, Sapienza Università degli Studi di Roma, IT
LUIGI CINQUE, Sapienza Università degli Studi di Roma, IT
DONATO CRISOSTOMI, Sapienza Università degli Studi di Roma, IT
GIAN LUCA FORESTI, Università degli Studi di Udine, Italy, IT
FABIO GALASSO, Sapienza Università degli Studi di Roma, IT
MARCO RAOUL MARINI, Sapienza Università degli Studi di Roma, IT
ALESSIO MECCA, Sapienza Università degli Studi di Roma, IT
DANIELE PANNONE, Sapienza Università degli Studi di Roma, IT

Deep Learning approaches have recently raised the bar in many fields, from Natural Language Processing to Computer Vision, by leveraging large amounts of data. However, they could fail when the retrieved information is not enough to fit the vast number of parameters, frequently resulting in overfitting and, therefore, in poor generalizability. Few-Shot Learning aims at designing models which can effectively operate in a scarce data regime, yielding learning strategies that only need few supervised examples to be trained. These procedures are of both practical and theoretical importance, as they are crucial for many real-life scenarios in which data is either costly or even impossible to retrieve. Moreover, they bridge the distance between current data-hungry models and human-like generalization capability. Computer Vision offers various tasks which can be few-shot inherent, such as person re-identification. This survey, which to the best of our knowledge is the first tackling this problem, is focused on Few-Shot Object Detection, which has received far less attention compared to Few-Shot Classification due to the intrinsic challenge level. In this regard, this review presents an extensive description of the approaches that have been tested in the current literature, discussing their pros and cons, and classifying them according to a rigorous taxonomy.

CCS Concepts: • **Computing methodologies** → **Visual inspection**; **Object detection**; **Object recognition**; **Object identification**; **Matching**.

1 INTRODUCTION

Over the past years, it is possible to witness a paradigm shift in most Machine Learning (ML) fields. The dominant pipeline that revolved around hand-crafted features is now often replaced by an end-to-end framework, in which a mapping from input to output is learned along with the most convenient representation of the input. These models usually have layered architectures, where deeper layers operate over more complex representations

Authors' addresses: Simone Antonelli, Sapienza Università degli Studi di Roma, Rome, IT, antonelli.1753685@studenti.uniroma1.it; Danilo Avola, Sapienza Università degli Studi di Roma, Rome, IT, avola@di.uniroma1.it; Luigi Cinque, Sapienza Università degli Studi di Roma, Rome, IT, cinque@di.uniroma1.it; Donato Crisostomi, Sapienza Università degli Studi di Roma, Rome, IT, crisostomi@di.uniroma1.it; Gian Luca Foresti, Università degli Studi di Udine, Udine, Italy, IT, gianluca.foresti@uniud.it; Fabio Galasso, Sapienza Università degli Studi di Roma, Rome, IT, galasso@di.uniroma1.it; Marco Raoul Marini, Sapienza Università degli Studi di Roma, Rome, IT, marini@di.uniroma1.it; Alessio Mecca, Sapienza Università degli Studi di Roma, Rome, IT, mecca@di.uniroma1.it; Daniele Pannone, Sapienza Università degli Studi di Roma, Rome, IT, pannone@di.uniroma1.it.

0360-0300/2022/2-ART $15.00
https://doi.org/10.1145/3519022

expressed in simpler representations obtained from the previous layers. However, the depth of these models comes at a cost: their expressivity results in extensive hypothesis spaces, which in turn require large amounts of data to learn a reliable hypothesis. In fact, training these models over a limited dataset may end up in an overfitted representation of the problem, that also includes non-informative features. Such features appear in the dataset but are not representative of the overall distribution of the samples. This reduces the generalizability of the models and hinders the performance over unseen samples. A peculiar characteristic of human intelligence is its ability to generalize from a few examples, e.g., children can learn how to play games after just a handful of rounds or new words with just a single example [6]. Contrarily, ad-hoc deep neural networks need to be trained for a long time [23] to obtain satisfying results. To bridge this gap between current ML models and human-like intelligence, *Few-Shot Learning* (FSL) aims at designing models which can successfully operate in a limited data regime. Analogously to humans, a few-shot model must leverage some kind of prior knowledge to be able to learn reliable hypotheses. In a *Few-Shot Classification* (FSC) task, for example, the model is expected to learn how to discriminate an object after K samples, for some small K (usually less than 30). In the literature, a problem in which a learner is tasked to recognize a category with no supervised samples is called Zero-Shot. Supervised information must come in different modalities to allow the model to learn, e.g., in the form of word embeddings, analogously to the case of a child that learns a visual task from a textual description. The ability to learn how to predict from a few examples is particularly desirable for Computer Vision (CV) systems, as the distributions of object categories in existing datasets are usually long-tailed, with most of the classes only having few supervised samples. In fact, many objects naturally have scarce examples or their annotations could be hard to obtain, e.g., the detection of rare forms of cancer when a few or no examples are available. Moreover, some important CV tasks are inherently few-shot: for example, in person search, the learner must be able to re-identify and localize a person in a gallery of images after having seen just a handful of photos. This task has important applications in video surveillance, allowing to track specific elements in a scene (e.g., criminal suspects or lost persons). For these reasons, many approaches tackling FSC have been proposed in recent years [15, 36, 38, 46, 48–50]. An element in common among many of these approaches is `Meta-Learning`. In particular, `Distance Metric Learning` methods are employed in [48, 50], while in [15, 38] prior knowledge influences the initialization of the model.

Few-Shot Object Detection (FSOD) nevertheless has received far less attention, showing a high complexity due to the localization part. Thus, it is impossible to directly use existing FSC approaches. Therefore, they must be adapted for the localization of specific objects. In general, as the data is not enough to provide a reliable hypothesis, a few-shot model must leverage prior knowledge. Since FSOD is a FSL task, this review follows the same taxonomy proposed in [56], showing a classification of few-shot approaches based on the different ways they exploit prior knowledge:

- **Data**. These methods use prior knowledge to augment the training dataset. Conventional models can then be trained on the augmented data.
- **Model**. These methods use prior knowledge to constrain the complexity of the hypothesis space, reducing its dimension. In this smaller space, the training dataset is sufficient to allow the system to effectively learn and provide a reliable hypothesis.
- **Algorithm**. These methods use prior knowledge to search for the parameters set which yields the best hypothesis. Prior knowledge alters the search strategy by providing a good initialization or guiding the search steps.

Concerning a more specific task, only certain branches of the broader taxonomy proposed in [56] have been explored. For this reason, this survey considers 4 different families of approaches: (i) `Data Augmentation`; (ii) `Transfer Learning`; (iii) `Distance Metric Learning` (DML); (iv) `Meta-Learning`. Section 2.4 details each category and relates them to the broader taxonomy in [56].

Based on these considerations, the survey continues as follow: Section 2 sketches the common background on the relevant topic; Section 3 presents the evaluation metrics, the public datasets, and the most used benchmarks; Section 4 describes the surveyed proposals according to a rigorous taxonomy; Section 5 mentions some relevant works about the Zero-Shot paradigm; Section 6 shows and briefly discusses a schematic overview of the presented proposals; Section 7 exposes open issues and the future research lines of FSOD; while Section 8 concludes the paper.

2 BACKGROUND

To provide a common background, Section 2.1 presents an overview of Object Detection (OD) in a broader context; Section 2.2 formalizes the FSL paradigm; Section 2.3 introduces the FSOD task, also depicting the basics of both Zero-Shot Learning (ZSL) and Zero-Shot Object Detection (ZSOD); and Section 2.4 provides a general purposes' description about `Data Augmentation`, `Transfer Learning`, `Distance Metric Learning`, and `Meta-Learning`.

2.1 Object Detection

This section aims at introducing the readers to the topic with some basic concepts. A comprehensive overview of OD approaches that were employed in the last two decades can be found in [31], while [21, 67] provide two reviews of deep learning-based approaches for the task.

OD is a hot-topic in CV, merging classification and localization in a single unified task. The goal of the former is to assign a class label to an image, whereas the latter aims at drawing a bounding box around each object of interest. A bounding box is usually defined by a starting point together with the height and width of the box. The main issue of OD is that the number of objects can vary for each input image. For this reason, early OD approaches were usually formulated as sliding window classification problems, assuming the presence of only one object for each region within the window. However, with the rise of deep learning, CNN-based methods have become the dominant OD solution. In a typical OD setting, a dataset of N_s supervised samples $D = \left(\{\mathbf{X}\}_{i=1}^{N_s}, \{y\}_{i=1}^{N_s}\right)$ is provided, where each $\mathbf{X}_i \in \mathbb{R}^{W \times H \times 3}$ is an image of height H and width W and each $y_i = \{(b_j, c_j)\}_{j=1}^{o_i}$ is a set of object annotations, each composed of a bounding box b_j and a class label c_j for each of the o_i objects contained in the image. Given an input image $\mathbf{X}_i$, modern models usually first extract a feature map $\mathbf{F} \in \mathbb{R}^{W \times H \times M}$, where M denotes the number of channels. The model should be able to identify an object in the sub-regions of the feature map, also called Regions of Interest (RoIs), and for each of them, perform two tasks: regression, in order to perfectly fit a bounding box around the object, and classification, to assign the correct label to the object in the bounding box. Depending on whether the region proposals are explicitly generated or not, these approaches can be divided into two categories. *Two-stage* detectors, also called proposal-based, firstly extract class-agnostic RoIs, and then refine and classify the proposed regions. Differently, *one-stage* detectors, also called proposal-free, directly aim at locating and classifying the objects in a single stage without the explicit generation of region proposals. The former nevertheless usually yield better results, as the region-proposal module can filter out many negative locations, facilitating the subsequent detection task. The latter are the most recent, and since they do not require a per-region classifier, they are conceptually simpler and significantly faster. All the approaches that are further treated leverage one or both paradigms, therefore these are detailed in the following part of this section. Two-stage detectors are explored first as they lay the foundations for one-stage ones.

2.1.1 Two-stage. The most widely adopted two-stage architecture is the `Faster R-CNN`, shown in Figure 1. It consists of a Region Proposal Network (RPN) and a detection network (`Fast R-CNN`) [17] that uses the proposals to detect objects. These networks share a backbone convolutional network, which provides convolutional feature maps. The RPN considers B reference boxes per location, also called anchor boxes, of different scales and aspect ratios. It then identifies region proposals by predicting the probability of an anchor to be foreground or background.

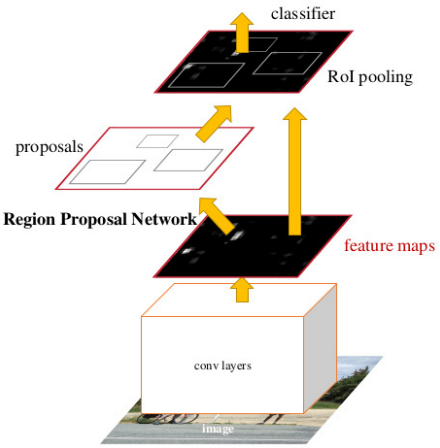

Fig. 1. The `Faster R-CNN` architecture: an input image is fed through a backbone CNN to output a set of feature maps; a RPN identifies RoIs that are pooled to obtain a fixed-size feature map for each proposal. A detection network predicts the class for each RoI and regresses the bounding box offsets. Figure from [41].

These anchors are refined to tightly bound the object. The region proposals are reshaped to extract a fixed-length feature for each of them to feed the detection network. The latter predicts the object class via a softmax classifier and produces the bounding box offsets via per-class bounding box regressors.

Region Proposal Network: In order to train the RPN, a binary class label object/non-object is assigned to each anchor. The anchor(s) with the highest Intersection over Union (IoU) with a ground-truth box and those which have an IoU overlap greater than 0.7 are deemed positive, while those with IoU lower than 0.3 for all ground-truth boxes are deemed negative. All the other anchors are ignored. Given an image with m anchors, its loss function is defined as:

$$\mathcal{L}\left(\{p\}_{i=1}^{m},\{t\}_{i=1}^{m}\right)=\frac{1}{N_{\text{obj}}}\sum_{i}\mathcal{L}_{\text{obj}}\left(p_i,p_i^*\right)+\lambda\frac{1}{N_{\text{loc}}}\sum_{i}p_i^*\mathcal{L}_{\text{loc}}\left(t_i,t_i^*\right) \tag{1}$$

where p_i is the predicted probability of anchor i to be an object, p_i^* is the corresponding ground-truth, t_i is a vector containing the parameterized coordinates of the predicted bounding box, and t_i^* is the same vector corresponding to the ground-truth box. N_{obj} and N_{loc} are respectively the batch size and number of anchor locations, and act as normalization factors. The classification loss $\mathcal{L}_{\text{obj}}$ is the binary cross-entropy, while the regression loss $\mathcal{L}_{\text{loc}}$ is a smooth ℓ_1, considered only for positive anchors. The latter behaves like ℓ_1-loss when the absolute value of the argument is high, and it behaves like ℓ_2-loss when the absolute value of the argument is close to zero.

Predictor head: The predictor head contains two sibling output layers: the first consists of a fully-connected layer followed by a softmax to compute a class probability distribution for each RoI, while the second produces the bounding box offsets $t^c=\left(t_{\text{x}}^c,t_{\text{y}}^c,t_{\text{w}}^c,t_{\text{h}}^c\right)$, for each of the $c=1,\ldots,N$ categories. The training RoIs have a ground-truth class c and a ground-truth bounding-box regression target t_{true}. The model outputs a tuple of predicted bounding offsets t_{pred}^c. The loss used for the predictor head train is the sum, for all the ROIs, of the following contribution:

$$\mathcal{L}\left(p,c,t_{\text{pred}}^c,t_{\text{true}}\right)=\mathcal{L}_{\text{cls}}(p,c)+\mathcal{L}_{\text{loc}}\left(t_{\text{pred}}^c,t_{\text{true}}\right) \tag{2}$$

where $\mathcal{L}_{\text{cls}}$ is the cross-entropy loss and the second task loss $\mathcal{L}_{\text{loc}}$ is a smooth ℓ_1.

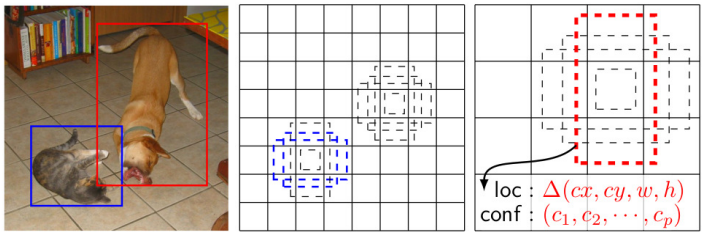

Fig. 2. SSD architecture: the input image is divided into an $S \times S$ grid, with different S for different scales. For each cell, a set of anchor boxes of different aspect ratios and scales is considered. The network is trained to predict bounding box offsets and confidences for every class for each of the anchor boxes.

2.1.2 One-stage. The two one-stage architectures mostly used as base detectors for FSOD models are You Only Look Once (YOLO) [39] and Single Shot Detection (SSD) [32]. They are detailed in the following, indicating their drawbacks and the possible improvements for countering those issues.

You Only Look Once (YOLO): YOLO exploits a single convolutional network to perform class and bounding box predictions using a limited set of candidate regions. To this aim, YOLO divides an image into an $S \times S$ grid, predicting for each cell C class probabilities, B bounding boxes and a confidence score for each of these. YOLO leverages contextual information about object classes from the whole image to detect less false positives in the background. However, this approach is limited by the assumption that each cell may contain a single object, and the coarse grid division may cause YOLO to miss small objects. To overcome this issue, YOLOv2 [40] introduces, among the other improvements, anchor boxes and multi-scale training.

Single Shot Detection (SSD): SSD improves YOLO by using anchor boxes adjusted on various object shapes. Moreover, SSD performs detection over multiple scales by leveraging multiple feature maps, so to make each of them responsive to different object scales. Figure 2 shows the SSD pipeline.

2.2 Few-Shot Learning

The current section aims at providing an introduction to FSL. FSL is a specific branch of ML in which a model is learned from limited supervised training information. The importance of FSL arises from the abundance of scenarios in which large amounts of data are not available, as collecting data is often costly and sometimes even impossible (e.g., dealing with rare diseases). Moreover, some problems are inherently defined as few-shot tasks: in person re-identification, for example, the input typically consists of the few frames, usually captured by surveillance cameras, containing the subject(s) of interest. Furthermore, many applications require direct input from the end-user (e.g., face or fingerprint recognition in mobile phones). In order to mitigate the user burden, the number of required samples should be as small as possible. As previously stated, FSL has also attracted a lot of attention from a more theoretical point of view, as humans exhibit an impressive capacity in solving problems with few input samples. This consideration makes FSL a necessary step to tighten the gap between generalization capability of humans and computers. Given a few-shot task with N classes, the training dataset $D_{\text{train}} = \{(\{x\}_{i=1}^{K}, y_j)\}_{j=1}^{N}$ consists of K examples for each of the N labels. For instance, considering a few-shot image classification problem, D_{train} consists of K images for each of the N classes. This kind of task is also known as N-way-K-shot classification. In general, the number of samples K gives the name to the few-shot task. Special K-shot tasks are One-Shot Learning, in which only one example with supervised information is available, and ZSL, in which no supervised information is provided. Another related topic is domain adaption, in which a model trained in a source domain should be able to tackle the same task in a different target distribution. This differs from FSL, where the source task and the target task should be different. An example of a work tackling this

task in a few-shot scenario is described in [52]. FSL approaches have to combine the low amount of information retrievable from the few input samples with some prior knowledge. The latter can be coarsely defined as any information the learner has about the function before seeing supervised samples. A thorough treatment of this topic is presented in [56].

2.3 Few-Shot Object Detection

FSOD is notably harder than FSC due to the localization task. The latter makes FSC approaches not directly applicable to the considered problem. In general, in a K-shot OD task, the model has to learn how to localize and classify objects belonging to the classes of interest with only K samples for each class as supervised data. Nevertheless, there are two different ways to determine whether an image is a valid sample: for some authors, it is sufficient that an image contains an object of the class of interest, while for others the image also does not have to contain any other object of the same class. In the first case, some objects may be overrepresented by appearing many times in the same image, while in the second one, the model is only allowed to see K instances of each class.

Given the success achieved in FSC tasks [50], some authors suggest only using few examples per class already at training time to mimic the test phase. This is achieved by splitting the training dataset in several N-way K-shot episodes: each of them is obtained by first sampling a subset of N classes, and then sampling K instances belonging to those classes for the support set and a certain number for the query set. Some details of this setup are introduced in Section 4.

2.3.1 Zero-Shot Learning and Zero-Shot Object Detection. In general, ZSL and ZSOD can be considered as boundary cases of FSL and FSOD. Unlike the latter, in ZSL and ZSOD no samples of specific classes are present. In fact, the purpose of ZSL and ZSOD is to predict (for the first) and to detect (for the second) new classes without having samples of them. Such novel classes are known as *unseen* classes, while those occurring in the training set are known as *seen* classes [9]. Formally, we can define $D_{train} = \{(x, y)|x \in X, y \in Y\}$ as the training dataset, where the tuple (x, y) consists of the sample x and the corresponding class y, with X and Y corresponding to the known samples set and the known classes set, respectively. Differently, the test dataset can be defined as $D_{test} = \{(t, z)|t \in T, z \in Z\}$, where t is a sample belonging to an unseen class z, with T and Z representing the set of samples belonging to unknown classes with the corresponding unknown classes set, and $Y \cap Z = \emptyset$. Other sources of information are usually exploited to get knowledge of the *unseen* classes such as taxonomies, textual descriptions, and visual annotations by involving other ML methodologies such as Natural Language Processing (NLP) to extract information from textual data. These strategies make ZSL and ZSOD the closest learning and detection approaches, respectively, to the human brain's capability of abstraction and understanding.

2.4 Few-Shot Object Detection Methods Taxonomy

This section introduces the categories of approaches that have been employed in FSOD. Their affiliation to the broader taxonomy suggested in [56] is explained by describing which aspect they improve by leveraging prior knowledge.

`Data Augmentation`. `Data Augmentation` is usually performed as a preprocessing step. Approaches belonging to this category try to augment the training dataset to make it large enough to provide reliable hypotheses to the model. To this aim, they add new artificial samples obtained by transforming the existing ones. However, in some cases, the dataset can be directly augmented at training time by integrating some augmenting mechanism in the architecture of the learning model. Some popular image transformations are rotation, shift, scaling, and flipping. Those operations are task-dependent: the same transformation may create meaningful samples for a certain task

while resulting in unrealistic data distribution in another. For this reason, ad-hoc transformations are usually employed, hindering the re-use over different tasks and datasets.

`Transfer Learning`. A standard `Transfer Learning` problem consists of two related tasks, one named source and the other target. The goal of the paradigm is to leverage knowledge learned over the source task to improve the performance of the learner over the target task. In the typical setting, the source task has a large-scale training dataset, while the target one suffers from data scarcity. Formally, given a source domain $\mathcal{D}_S$ and its learning task $\mathcal{T}_S$, a target domain $\mathcal{D}_T$ and its learning task $\mathcal{T}_T$, `Transfer Learning` aims at improving the learning of the target predictive function $f_T(\cdot)$ in $\mathcal{D}_T$ exploiting the knowledge retrieved through $\mathcal{D}_S$ and $\mathcal{T}_S$ [30].

While this technique has been widely used in FSC, it is harder to apply it to FSOD: the backbone networks employed in typical OD architectures are usually pre-trained classifiers that need to be trained on a significant dataset to remove the gap between detection and classification. Moreover, deep detectors are more prone to overfitting than deep classifiers due to the need of a consistent number of object-specific representations in order to also address the localization [8].

`Distance Metric Learning`. Another widely used approach in CV is DML. It aims at embedding each sample to a lower-dimensional space such that similar samples are close together, while dissimilar ones can be discriminated in an easier way. Intuitively, a lower-dimensional space allows employing a smaller hypothesis space, which in turn allows the model to be effectively trained with fewer instances. In general, embedding learning approaches employ an embedding function f for the test samples, an embedding function g, (possibly different from f) for the training samples, and a similarity function $s(\cdot,\cdot)$ which measures the similarity between $f(x_{\text{test}})$ and the considered training sample $g\left(x^{(i)}_{\text{train}}\right)$ in the embedding space [56].

`Meta-Learning`. `Meta-Learning` is a sub-field of ML aiming at learning how to learn. Ideally, given a distribution of learning tasks, a *meta-learner* should be able to gradually learn generic information across the tasks to use it to modify the algorithm that effectively solves them. Formally, given a task distribution $p(\mathcal{T})$, learning how to learn is equivalent to obtaining the meta-knowledge ω that results in the minimum expected loss over $p(\mathcal{T})$:

$$\min_{\omega} \mathop{\mathbb{E}}_{\mathcal{T}\sim p(\mathcal{T})} \mathcal{L}(\mathcal{D}_{\mathcal{T}};\omega),$$

where $\mathcal{D}_{\mathcal{T}}$ is the dataset related to the learning task $\mathcal{T}$ and $\mathcal{L}$ is the error obtained by solving task $\mathcal{T}$ with the meta-knowledge ω. The meta-learner is trained over a set of source tasks $\mathcal{T}_{\text{source}}$ sampled from $p(\mathcal{T})$, each having its training and test data. The "learning how to learn" is given by:

$$\omega^* = \arg\min_{\omega} \sum_{\mathcal{T}\in\mathcal{T}_{\text{source}}} \mathcal{L}\left(\mathcal{D}_{\mathcal{T}};\omega\right)$$

As for any learning model, a testing phase is required; in meta-testing, another set of tasks $\mathcal{T}_{\text{target}}$, disjoint from $\mathcal{T}_{\text{source}}$, is sampled from $p(\mathcal{T})$. These are used to test the generalization ability of the meta-learner, using the loss averaged across the tasks in $\mathcal{T}_{\text{target}}$ as the meta-testing error.

3 EVALUATION

This section aims at establishing the necessary tools for the evaluation of FSOD models. In particular, Section 3.1 introduces the most adopted metrics, Section 3.2 describes existing datasets used for the task, and Section 3.3 attempts to collect the different learning setups used in literature.

3.1 Metrics

An OD model requires an evaluation metric to measure the effectiveness of a predicted bounding box. A possible parameter is provided by the *Intersection over Union* (IoU) that analyzes the overlaps of the predicted bounding boxes with those provided by the ground-truth. It is computed as:

$$\text{IoU} = \frac{\text{area of overlap}}{\text{area of union}} = \frac{b_{\text{true}} \cap b_{\text{pred}}}{b_{\text{true}} \cup b_{\text{pred}}} \tag{3}$$

where b_{true} and b_{pred} are the ground truth and predicted bounding box respectively. The introduced metric can be used to define a contigency matrix. In particular, given a confidence threshold τ_{c}, an IoU threshold τ_{IoU} and a set of ground-truth annotations G, the contingency matrix can be computed with Listing 1.

```
for each detection (b_pred, c_pred, σ) that has a confidence score σ > τ_c:
    O_c ← {o ∈ G | o has class c_pred}
    b_true ← argmax_{o∈O_c} IoU(b_pred, o_pred)
    if O_c = ∅ or IoU(b_pred, b_true) < τ_IoU:
        the detection is a false positive
    else:
        the detection is a true positive
```

Listing 1. Contigency table computation.

Leveraging Listing 1, *true positives (TP)* are detections for which (i) the confidence score > τ_{c}; (ii) the predicted class matches the class of a ground-truth; (iii) the predicted bounding box has an IoU greater than τ_{IoU} with the ground-truth. Detections violating one of the last two conditions are flagged as *false positives (FP)*. Differently, if a valid detection violates the first condition, it is considered a *false negative (FN)*. Finally, detections that achieve a low confidence score are flagged as *true negatives (TN)*; these, however, are not very informative, as they represent for the vast majority of the predictions, and therefore are usually ignored in practice. With these redefinitions, it is possible to use the standard *precision* and *recall* metrics.

Different pairs made up by precision and recall are obtained by setting different values for the confidence score threshold. These pairs are exploited to plot a *Precision-Recall curve*, which visualizes how the two metrics are related. Likewise, different pairs made up by IoU and recall are obtained by changing τ_{IoU}. These can be used to draw a *IoU-Recall curve* that helps in evaluating the effectiveness of detection proposals. The *Precision-Recall curve* can be used to compute the *Average Precision* (AP) by averaging the precision across all recall levels. The latter averaged over all the classes yields the *mean Average Precision* (mAP). The *Average Recall* (AR) is the recall averaged over all $IoU \in [0.5, 1.0]$. As for AP, *Mean Average Recall* (mAR) is the corresponding averaged version.

3.2 Datasets

Annotated datasets are used as benchmarks to provide a fair comparison among different algorithms and architectures. Furthermore, the growth in complexity, size, and annotation number increases the challenge level, encouraging in constant development of new and improved techniques. The most used datasets for the task of FSOD are: (i) `Pascal VOC` (in brief `VOC`) [11, 12], (ii) `MS-COCO` [29], (iii) `LVIS` [18], and (iv) `ImageNet` [10]. `Pascal VOC` and `MS-COCO` share their name with the corresponding detection challenges, while `ImageNet` is used in the *ImageNet Large Scale Visual Recognition Challenge* (`ILSVRC`) [44], named in this paper as `ImageNet 2015`. This section aims at providing a brief overview of the datasets employed in the considered approaches. The characteristics of each dataset are summarized in Table 1, while their statistics are reported in Table 2.

The `Pascal VOC` challenge has been ongoing from 2005 to 2012 and offered a dataset suited for classification, segmentation, and OD. However, FSOD tasks usually employ `VOC07`, `VOC10` and `VOC12`. Containing 20 object

Table 1. The main characteristics of the considered datasets.

Dataset	Classes	Objects per image	Image size	Started year
Pascal VOC	20	2.4	470 × 380	2005
ImageNet 2015	200	1.5	500 × 400	2009
MS-COCO	91	7.3	640 × 480	2014
LVIS	1000+	11.2	-	2019
FSOD	1000	2.82	varying	2020

Table 2. Statistics of the commonly involved datasets in FSOD.

Dataset	Number of images			Number of object annotations	
	train	val	test	train	val
VOC07	2501	2510	4952	6301	6307
VOC12	5717	5823	10,991	13,609	13,841
ImageNet 2015	456,567	20,121	51,294	478,807	55,502
MS-COCO	82,783	40,504	81,434	604,907	291,875
LVIS	100,170	19,809	19,822	1,270,141	244,707
FSOD	52,350	14,152	-	147,489	35,102

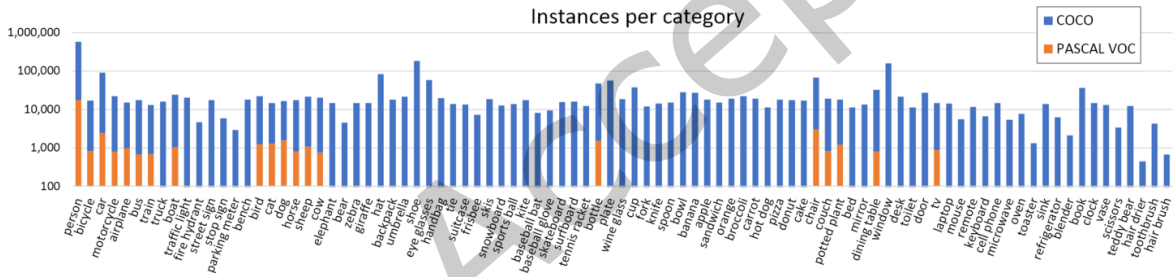

Fig. 3. Distribution of the categories in MS-COCO. Image from [29].

classes and consisting of only ≈ 10k images, Pascal VOC has been gradually supplanted by larger datasets, such as ImageNet 2015, which contains 200 object categories in ≈ 500k images. The categories were carefully selected considering different factors such as object scale, level of image clutteredness, and average number of object instances. For a more realistic test environment, some of the test images contain none of the 200 categories. Nevertheless, objects in ImageNet 2015 are often still unrealistically well-positioned and scaled. To account for this issue, in MS-COCO the authors gathered images of complex everyday scenes, containing common objects in their natural context and forcing the observer to understand the scene in order to successfully recognize objects. It should be noted that the authors did not waive the dataset scale in the process, as it consists of ≈ 330k images, of which ≈ 200k labeled, resulting in ≈ 1.5 million object instances belonging to 80 classes. Figure 3 shows a graphical representation of class distributions in MS-COCO. Furthermore, LVIS enriches MS-COCO with annotations regarding ≈ 2 million objects for over 1000 categories. Exhibiting a long tail of categories with few training samples, LVIS is particularly suited for FSOD tasks. It is also worth mentioning the FSOD dataset, released in [13]. It contains 1000 object categories specifically designed for FSL and for evaluating the generality of a model on novel categories. The dataset contains around 66k images and 182k bounding boxes.

3.3 Benchmarks

Different authors selected different metrics, datasets, and even methodologies to evaluate their models. This section aims at formalizing the most widely adopted ones. Instead, the less common ones will be described in the works where they have been employed.

The two most adopted benchmarks have been introduced in [22]. The first is based on VOC07 and VOC12 datasets. In the presented setting, the model is trained on $\texttt{VOC07}_{\text{train}} \cup \texttt{VOC12}_{\text{trainval}}$ and tested upon $\texttt{VOC07}_{\text{test}}$ using mAP metric. Out of the 20 object categories included in VOC, 5 are randomly sampled to be used as novel classes, while the remaining 15 are used as base classes. The model is then evaluated with 3 different base/novel splits. During base training, only annotations of the base classes are given. For few-shot fine-tuning, each class of objects only has K annotated bounding boxes, with $K = 1, 2, 3, 5, 10$. In the following, we refer to this benchmark as VOC-07/12. The second benchmark, instead, leverages MS-COCO, training the model over $\texttt{COCO}_{\text{train}} \cup \texttt{COCO}_{\text{val}^*}$, where $\texttt{COCO}_{\text{val}^*}$ is a subset of 35K images of $\texttt{COCO}_{\text{val}}$ and testing it on $\texttt{COCO}_{\text{val-val}^*}$, which consist of the remaining $5k$ images of $\texttt{COCO}_{\text{val}}$. Out of the 80 object classes, the 20 classes which overlap with VOC are used as novel classes, while the remaining 60 are used as base classes. Unlike the previous benchmark, different versions of AP and AR are presented as a metric (more detail in Section 6). In the following, we refer to this benchmark as MS-COCO. Finally, some works employ a cross-domain evaluation setting in which the model is trained on all the 60 base classes of MS-COCO and evaluated on VOC. We will refer to this benchmark as MS-COCO→VOC.

4 FEW-SHOT OBJECT DETECTION APPROACHES

As previously introduced, FSOD approaches can be divided into four families (see Section 2.4). Therefore, this section describes the existing approaches, dividing them accordingly.

4.1 Data Augmentation

Data Augmentation, which aims at inflating the size of a dataset, is a commonly used solution to the problem of having few samples. In general, the majority of Data Augmentation approaches apply a transformation to already existing data. However, when a few data is available, Data Augmentation may result very difficult if not impossible. The methods in this category try to exploit prior knowledge to augment the training dataset so that it should be enough to learn a reliable hypothesis.

4.1.1 Multi-Scale Positive Sample Refinement for Few-Shot Object Detection (MPSR) [58]. The authors tackle the problem of scale variation by generating multi-scale positive samples (e.g., object pyramids) and refining the prediction at various scales.

Architecture: The proposed model is based on the Faster R-CNN but substitutes the backbone with a Feature Pyramid Network [28] and exploits a class-agnostic regression loss in the detection head. To add multi-scaled samples, each object is independently extracted and resized to various scales, denoted as object pyramids. Nevertheless, in standard object pyramids, each image only contains a single instance, which is inconsistent with the standard detection pipeline; therefore, an extra positive sample refinement branch is introduced to adaptively project the object pyramids into the standard detection network. Each input image is processed by a Faster R-CNN branch, while an independent object contained in the image is provided to the refinement branch. As shown in Figure 4, the cropped object is then resized to various scales, and, for each of those, specific feature maps from the FPN are selected to be fed to both the RPN head and the detection head for refinement. In the MPSR branch, the outputs of the RPN and the detection heads also include objectness scores and class-specific scores. Given an image, the loss function of the RPN head containing the Faster R-CNN and the MPSR branch is

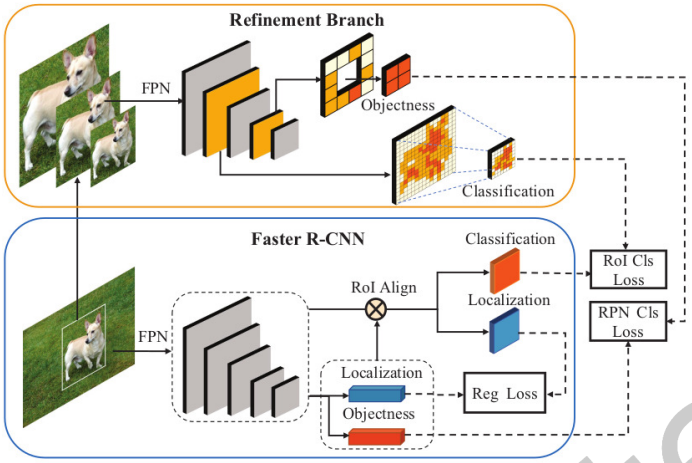

Fig. 4. *MPSR* [58] architecture. A `Faster R-CNN` branch processes the input image. The detected cropped object is supplied to the Refinement Branch that resizes it into various scales and feeds them into a FPN. Feature maps (yellow rectangles) are selected to compute the objectness and perform the classification, so to add the corresponding losses to the `Faster R-CNN` loss. Figure from [58].

defined as:

$$\mathcal{L}_{\text{RPN}} = \frac{1}{N+M}\sum_{i=1}^{N+M}\mathcal{L}_{\text{obj}}^{i} + \frac{1}{N}\sum_{i=1}^{N}\mathcal{L}_{\text{loc}}^{i}$$

where the only difference from the `Faster R-CNN` loss in Equation (1) is the new M selected positive anchor samples for refinement. Instead, the loss function of the detection head is defined as:

$$\mathcal{L}_{RoI} = \frac{1}{N_{\text{RoI}}}\sum_{i=1}^{N_{RoI}}\mathcal{L}_{\text{cls}}^{i} + \lambda\frac{1}{M_{\text{RoI}}}\sum_{i=1}^{M_{\text{RoI}}}\mathcal{L}_{\text{cls}}^{i} + \frac{1}{N_{\text{RoI}}}\sum_{i=1}^{N_{\text{RoI}}}\mathcal{L}_{\text{loc}}^{i}$$

where M_{RoI} is the number of selected RoIs in MPSR, while $\mathcal{L}_{\text{cls}}$ and $\mathcal{L}_{\text{loc}}$ are losses of the `Faster R-CNN` predictor head. After the training, the extra MPSR branch is removed and only `Faster R-CNN` with FPN is used for inference.

Results: The architecture was evaluated over the first two benchmarks presented in Section 3.3. The results over `VOC-07/12` exhibited a significant variability, being susceptible to the class split: the authors tried 3 different $15:5$ splits to pick the novel and base classes, respectively yielding a mAP of 41.70, 24.40, and 35.60 for the one-shot task. The achieved results are analogous also for $K = 3, 5, 10$. On average, the model respectively obtained a mAP of 44.30, 47.70, and 53.30 for these values of K. Regarding the `MS-COCO` benchmark, the model obtained a mAP of 9.80 and 14.10 for $K = 10, 30$. The authors also reported the `MS-COCO` metrics AP50 and AP75, over which the model yielded a score of 17.90, 25.40 for the first and 9.70, 14.20 for the latter.

4.2 `Transfer Learning`

As introduced in Section 2.4, in `Transfer Learning` a source task is used to improve the performance of a learning model over a target task, making the framework particularly suitable for FSL. In this specific context, the

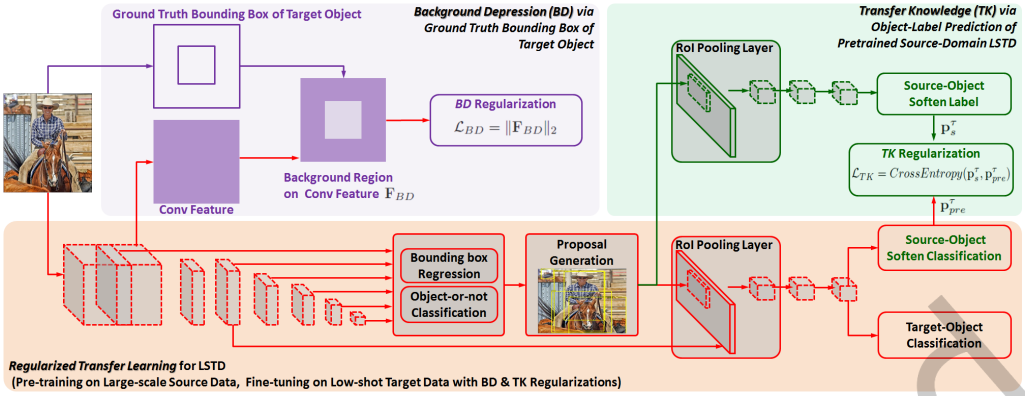

Fig. 5. The *LSTD* architecture. Red: the Low-Shot Transfer Detector. SSD-based bounding box regression is exploited for localization, while classification follows the Faster R-CNN model. Violet: the BD regularizer adds the ℓ_2-norm of the background to the loss. Green: the TK regularizer instills source-domain knowledge into the target domain. Figure from [8].

learner must exploit the prior knowledge from the data-abundant source task to be able to operate successfully in the target few-shot scenario.

4.2.1 LSTD: A Low-Shot Transfer Detector for Object Detection (LSTD) [8]. In [8], the authors suggested an architecture expressively designed to manage Transfer Learning difficulties in few-shot detection by leveraging both the SSD and the Faster R-CNN.

Architecture: Bounding box regression is performed according to SSD, identifying a set of default candidate boxes for each spatial location of the convolutional feature map. A smooth ℓ_1 loss is computed over positive candidate boxes to penalize the offsets error between the predicted and the ground-truth bounding boxes. Instead, object classification follows the Faster R-CNN model, except for the final two fully-connected layers. The latter are replaced with two convolutional layers to reduce overfitting. The main idea is that a coarse-to-fine classifier may help to manage the fine-tuning issues, compared to the $(N + 1)$ object classification that is used in SSD for each default box. In this way, the classifier can focus on the objects that the RPN has already detected: these should share more common characteristics between source and target ones than those in the background. Figure 5 shows an overall view of the proposed architecture. The authors also introduce two novel regularizers, namely *Transfer Knowledge (TK)* and *Background Depression (BD)*, to respectively leverage object knowledge from source and target domains. The former projects the ground-truth bounding boxes into the convolutional feature map and adds the ℓ_2-norm of these activations to the global loss. A feature heatmap is computed to show the influence of the background on the prediction over target objects. The model is focused on the suppression of the influence of the background regions in target objects. Differently, the latter exploits source-domain knowledge as a regularizer to fine-tune on the target-domain. An extra head is added to the target-domain *LSTD*, tasked with a classification over the classes of the source-domain. During training, an image of the target domain is fed to both the source-domain *LSTD* and the target-domain one, while both models use the RoIs produced by the target-domain model. The minimization of the difference between the class distribution output of the source-domain *LSTD* and the target-domain one is enough to instill the source-domain knowledge acquired by the first model into the second.

Results: The authors evaluated the architecture over three tasks, each having a large-scale training set in the source domain and a K-shot training set in the target domain, with $K = 1, 2, 5, 10, 30$. In the first task (T1), MS-COCO was used as the source domain while a subset of ImageNet 2015 with 50 selected classes was exploited

as the target domain. For the second task (T2), the source domain was a subset of MS-COCO, consisting of all the images that include objects belonging to a selected set of 60 classes, while the target domain was the test dataset from VOC07. In the third task (T3), 181 classes were chosen from ImageNet 2015 as the source domain, and the test dataset from VOC10 was used as the target. Object categories for source and target are non-overlapped and the mAP was used as a metric. The results shown that there is no linearity between the increase of K and the achieved performance. For example, in the case of T1, the results for each K was 19.20, 25.80, 37.40, 44.30, and 55.80 respectively. As can be seen, between $K = 1$ and $K = 2$ there is a difference of about 6 points, while between $K = 2$ and $K = 5$ it is only of about 11. This trend is even more evident in the case of $K = 10$ and $K = 30$. In fact, the difference among the results was considerably smaller compared with the increase of K. This consideration is also valid for the other works analyzed in this review.

Evolution of the proposal: The authors have extended this work by levering the proposed architecture to build a Progressive Object Transfer Detection (POTD) procedure to perform OD on weakly-annotated images [7]. In particular, POTD is composed of two main processes. The first relies on the learning of a warm-up detector based on *LSTD*. The second one, called Weakly-Supervised Transfer Detection (WSTD), is used to transfer the warm-up detector capability into an architecture that handles weakly-annotated images. Since weak supervision is not a topic covered by this survey, readers should refer to the original paper for further details.

4.2.2 Frustratingly Simple Few-Shot Object Detection (TFA) [54]. This work underlines that FSOD can be successfully tackled by employing a clever two-stage training scheme, which effectively fine-tunes a deep detector. To this end, it proposes to train a Faster R-CNN with the addition of a novel instance-level feature normalization to fine-tune it.

Architecture: While any deep detector can be integrated in the framework, a Faster R-CNN is used in this work. In the latter, backbone and RPN features may be considered class-agnostic, so the features learned from the base classes are likely to be transferred to the novel classes without fine-tuning. Instead, the representations learned by the box predictor are class-specific. They are not transferred from base to novel classes, and need to be fine-tuned over the latter. The fine-tuning framework shows a two-stage approach. In the former, the model is trained only on the base classes with the Faster R-CNN loss function. In the second stage, the box predictor is fine-tuned over a few-shot training set containing both base and novel classes. These predictors are randomly initialized. The same loss function is used, but with a smaller learning rate. Figure 6 shows an overview of the two proposed stages. For the Box Classifier, given an input feature matrix $\mathbf{X} \in \mathbb{R}^{d \times m}$, a common fully-connected classifier outputs a matrix $\mathbf{S}$ that encodes the similarity between input features and classes, each entry of this matrix is provided by: $s_{i,j} = \mathbf{x}_i^\top \mathbf{w}_j$, where $\mathbf{x}_i$ is the i-th object proposal and $\mathbf{w}_j$ is the per-class weight vector of the j-th class. The authors suggested scaling the similarity scores by taking instead the cosine similarity between each feature vector and class weight vector. This choice should reduce the intra-class variance encountered by the classifier, therefore mitigating the loss of detection accuracy that goes from base classes to novel classes.

Results: The authors evaluated the proposed approach on both the VOC-07/12 and MS-COCO benchmarks presented in Section 3.3, and proposed a new benchmark which leverages three datasets, namely VOC, MS-COCO and LVIS. The architecture was also tested with a fully-connected (FC) classifier to evaluate whether the proposed cosine similarity (CS) classifier is beneficial. Taking $K = 1$, the FC version yielded a mAP of 36.80, 18.20, and 27.70 for the three different splits on the VOC benchmark, while the CS version respectively yielded a mAP of 39.80, 23.50 and 30.80. Analogous results were achieved for the remaining values of K, indicating that the cosine similarity-based classifier is indeed advantageous for the task. Nevertheless, the difference in performance between the two architectures was less sharp in the MS-COCO benchmark: while the FC-based architecture respectively yielded a mAP of 10.00 and 13.40 for $K = 10$ and $K = 30$, the cosine similarity one results in 10.00 and 13.70, producing a negligible improvement. The new benchmark was introduced by the authors aiming at countering two issues: the neglect of the performances on base classes, as only the novel ones are considered

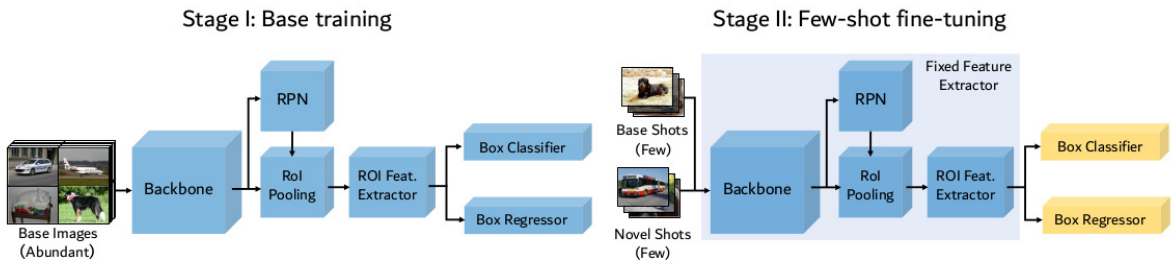

Fig. 6. *TFA* [54] architecture. A `Faster R-CNN` is trained over the data-abundant base classes and then fine-tuned over a small balanced few-shot set of novel and base classes.

in the previous evaluation benchmarks, and the large sampling variance in the few-shot training examples, which could confound the comparison of different models. To address the first issue, the AP on the base classes and on the novel classes is computed, in addition to the overall AP. For the second issue, the authors sampled different groups of few-shot training examples and ran the experiments multiple times, reporting an average of the obtained performances and a confidence interval. This evaluation protocol was implemented for the datasets `VOC`, `MS-COCO`, and `LVIS`. The setting used for the first two was the same shown in Section 3.3, except for the mentioned additions. The `LVIS`-based benchmark split the classes encountered in the `LVIS` dataset into three categories: frequent, common, and rare. The first two are used as the base classes, and the latter as the novel. To balance the dataset for the fine-tuning, up to 10 instances for each class are sampled from it. In this setting, both the FC and the CS architectures obtained an overall mAP of 41.80. Nevertheless, the CS one resulted again in a slight improvement in the mAP: it respectively achieved, for the rare, common, and frequent classes, a mAP of 17.30, 26.40, and 29.60, versus the 15.50, 26.00 and 28.60 achieved by the FC architecture.

4.2.3 Semantic Relation Reasoning for Shot-Stable Few-Shot Object Detection (SRR-FSD) [66]. This work underlines how the semantic relation between base and novel classes is constant regardless of data availability necessary to train an object detector. It introduces a semantic space built from the classes' word embeddings and trains the detector to project object instances onto the semantic space. In this way, the detector learns novel objects from visual and semantic information.

Architecture: The proposed architecture is built on top of the `Faster R-CNN`. The semantic space $\mathbf{W}_e \in \mathbb{R}^{N \times d_e}$, consists of a set of word embeddings of dimension d_e, one for each of the N classes, including the background class. The detector aims at learning the projection $\mathbf{P} \in \mathbb{R}^{d_e \times d}$ such that the feature vector $\mathbf{v} \in \mathbb{R}^d$ of a region proposal is aligned with the word embedding of its corresponding class. The idea is that the learned semantic projection is able to align concepts from the visual space and those from the semantic one. A knowledge graph $\mathbf{G}$ manages the relationships between classes. It is represented by an adjacency matrix $N \times N$ that, for each pair of classes, indicates their connection strength. Thus, in the classification subnet, the probability distribution over classes is obtained as: $\mathbf{p} = \text{softmax}(\mathbf{G}\mathbf{W}_e\mathbf{P}\mathbf{v} + \mathbf{b})$, where $\mathbf{b} \in \mathbb{R}^N$ indicates the bias vector to model unbalanced classes, and $\mathbf{G}$ is involved via the graph convolution operation. Moreover, $\mathbf{G}$ is dynamically learned by leveraging a self-attention architecture. In this way, the extension of the knowledge graph with new classes only requires the addition of the corresponding embeddings and to fine-tune the detector. Finally, similar to [54], the authors fine-tune the parameters of the last layers of the architecture. For the classification subnet, they fine-tune the relation reasoning module $\mathbf{G}$ and the projection matrix $\mathbf{P}$. Since the layer that extracts the feature vectors of the region proposals is shared between the classification and the localization subnets, the authors doubled this layer to fine-tune them independently. An overview of the architecture is depicted in Figure 7.

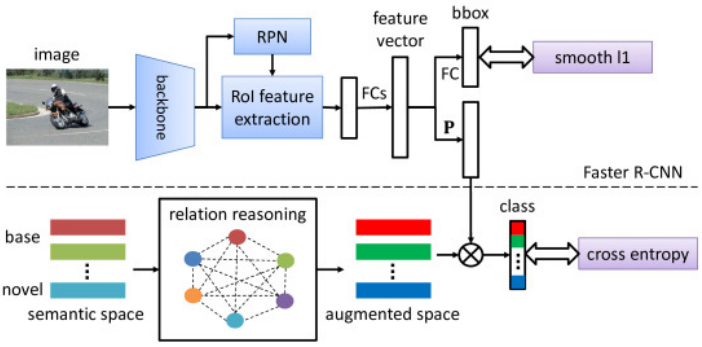

Fig. 7. *SRR-FSD* [66] architecture. In the bottom part, the construction of the semantic space is shown by using the word embeddings. This is then augmented via the relation reasoning module. While the query image is processed using the Faster R-CNN architecture, where before the classification, the visual features are projected onto the semantic space. Figure from [66].

Results: The architecture was benchmarked with VOC-07/12, MS-COCO, and MS-COCO→VOC. Furthermore, the authors introduced a new, more realistic benchmark where novel classes are removed from the classes on which the backbone network is pre-trained. In such a way, the backbone has no implicit access to large amounts of objects samples overlapping with the novel classes. In particular, the authors removed the classes that overlap with the novel sets of VOC-07/12. SRR-FSD is compared to [22, 54, 60]. In the VOC-07/12 benchmark, SRR-FSD demonstrated shot-stable results. In fact, it achieved competitive performances when $K = 5, 10$, whereas it outperformed the compared approaches when $K < 5$. For instance, considering $K = 3$, SRR-FSD achieved a mAP of 51.3, 39.1, and 44.3 for the Novel Set 1, 2, and 3, respectively. Instead, in the MS-COCO benchmark, SRR-FSD outperformed the other architectures with both $K = 10$ and $K = 30$, reaching an AP_{50} of 23.3 and 29.2 respectively. The same goes for the cross-domain benchmark MS-COCO→VOC, where the proposed architecture achieved a mAP of 44.5 with $K = 10$. Lastly, on the more realistic benchmark SRR-FSD outperformed the other methods in most settings, achieving a mAP of 52.6, 34.7, and 39.7 on the 3 different splits considering $K = 3$.

4.2.4 Universal-Prototype Enhancing for Few-Shot Object Detection (FSOD-UP) [57]. The idea behind this work is that class-prototypes, effective in FSC, are not fit for the task of FSOD. For this reason, it suggests learning general features from objects belonging to all the classes. These should encode intrinsic object characteristics and be transferable to objects from unseen classes. These universal features are used to enrich the specific object features before the prediction step.

Architecture: Also in this case, a Faster R-CNN is employed as the base detection model. The authors define the *universal prototypes* to be D vectors $\mathbf{c}_i \in \mathbb{R}^m$. These are obtained by feeding the convolutional feature map $\mathbf{F}$ to a convolutional layer: $\mathcal{I} = \mathbf{W} * \mathbf{F} + \mathbf{b}$, with $\mathbf{W} \in \mathbb{R}^{3\times3\times m\times D}$ and $\mathbf{b} \in \mathbb{R}^D$. The output descriptor for a universal prototype c_i is then given by:

$$\mathcal{V}_i = \sum_{j=1}^{wh} \frac{e^{\mathcal{I}_{j,i}}}{\sum_{i=1}^{D} e^{\mathcal{I}_{j,i}}} \left(\mathbf{F}_j - \mathbf{c}_i\right) \tag{4}$$

where $\mathbf{F}_j - \mathbf{c}_i$ is a residual operation. The concatenation of $\mathcal{V}$ and $\mathbf{F}$ is provided as the input for the RPN. As shown in Figure 8, these prototypes are also used to represent object-level features. To this aim, they are refined by taking the affine transformation: $\mathcal{A} = \alpha \odot C + \beta$, where $\alpha \in \mathbb{R}^{D\times1}$ and $\beta \in \mathbb{R}^{D\times1}$ are the transformed parameters, and $\odot$

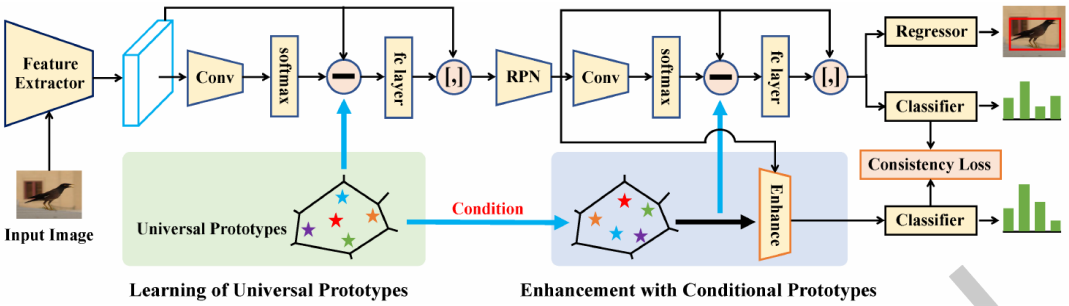

Fig. 8. *FSOD-UP* [57] architecture. A `Faster R-CNN` is modified by adding a set of layers between the CNN backbone and the RPN that computes universal prototypes. Conditional prototypes are computed from the RPN output and used for the prediction task. Figure from [57].

is element-wise product. The conditional prototypes $\mathcal{A} \in \mathbb{R}^{D \times m}$ are used as in Equation (4) to yield final output descriptors $O \in \mathbb{R}^{n \times D \times m}$. These are concatenated to the proposals P and fed to the classifier for the prediction task.

Results: The approach was evaluated over both `VOC-07/12` and `MS-COCO`, in comparison with [47, 59, 60]. In the former, the approach significantly outperformed the baselines for every K. In particular, it obtained a mAP of 50.3, 41.2, and 43.9 for $K = 3$ when considering Novel Sets 1, 2, and 3 respectively. In the latter, the approach obtained better performance with $K = 30$, achieving an AP of 15.6. Instead, with $K = 10$ it reached an AP of 11.0, underperforming [59] by a small amount.

4.2.5 Discussion. All the works in this section leverage a `Faster R-CNN` architecture. The latter is modified by [8] employing `SSD`-like bounding box regressors and by [66] enriching it with semantic information from the class word embeddings. While [8, 57] update all the parameters in the fine-tuning stage, the proposal in [54] shows that it might not be necessary to change the architecture at all if an adequate fine-tuning scheme is involved. The simplicity of the architectures makes `Transfer Learning` one of the most effective approaches for FSOD, as it manages to reach state-of-the-art results without the need for complex models and/or training frameworks.

4.3 `Distance Metric Learning`

DML is a widely used approach in CV and aims at embedding samples to a lower-dimensional space where the distances among them reflect their similarity. Intuitively, a lower-dimensional space allows to employ a smaller hypothesis space and consequently to effectively train the model with fewer instances. Following [56], metric learning is composed of: (i) an embedding function f for the test samples; (ii) an embedding function g for the training samples; (iii) a similarity function $s(\cdot, \cdot)$ which measures the similarity between the test sample $f(x_{test})$ and all the training samples $g(x_i)$ in the embedding space. For each x_{test}, the classification is performed by associating it with the class of the closest embedded training sample $g(x_i)$.

4.3.1 RepMet: Representative-based Metric Learning for Classification and One-Shot Object Detection (RepMet) [47]. In this approach, each class is represented by a mixture model with multiple modes so that their center can be considered the representative vector for the class. The idea is to jointly learn the backbone parameters, the embedding space, and the distribution of the training classes in that space, i.e., the representative vectors.

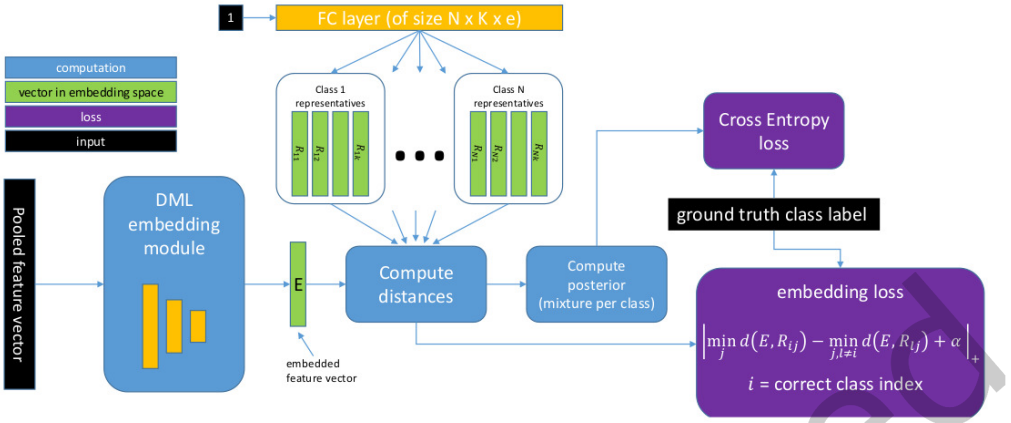

Fig. 9. *RepMet* [47] architecture. The K representative vectors are learned for each class. Given an input, the predicted class is provided by the representative vector closest to its DML embedding. A contrastive loss is employed to ensure a margin between the classes. Figure from [47].

Architecture: As depicted in Figure 9, the pooled feature vector $\mathbf{X} \in \mathbb{R}^f$ goes through a DML embedding module, which consists of a few fully connected layers employing ReLUs and batch normalization. This operation is performed to obtain a lower-dimensional vector $E \in \mathbb{R}^e$. In addition, a set of representative vectors R is also learned, where each $R_{ij} \in \mathbb{R}^e$ represents the center of the j-th mode of the learned mixture distribution in the embedding space for the i-th class. These, in particular, are obtained from the weights of a FC layer with scalar input 1. At this point, the distance matrix from the embedded feature vector E to the representative vectors R can be computed as $d_{ij}(E) = d(E, R_{ij})$. Assuming that all class distributions are mixtures of isotropic multi-variate Gaussians, the distance matrix can be exploited to compute the probability of the given RoI in each mode j of each class i: $p_{ij}(E) \propto \exp\left(-\frac{d_{ij}^2(E)}{2\sigma^2}\right)$. The choice of authors to upper-bound the class posterior with the following conditional probability allows them to avoid the learning of the mixture coefficients for the modes, that at test time becomes hard to perform: $\mathbb{P}(C = i \mid X) = \mathbb{P}(C = i \mid E) \equiv \max_{j=1,\dots,K} p_{ij}(E)$. Using this approach, there is no need to explicitly model the background probability, as it can be estimated via its lower-bound: $\mathbb{P}(\mathcal{B} \mid X) = \mathbb{P}(\mathcal{B} \mid E) = 1 - \max_{ij} p_{ij}(E)$. The model is trained exploiting a loss function composed of two distinct losses. The former is a regular cross-entropy with the ground truth labels given for the image (or RoI) corresponding to X. The latter is a triplet-loss that aims at forcing a margin of α between the distance of E to the closest representative of the correct class and the distance of E to the closest representative of a wrong class, and it is given by:

$$\mathcal{L}(E, R) = \left|\min_j d_{i^* j}(E) - \min_{j, i \neq i^*} d_{ij}(E) + \alpha\right|_+ \tag{5}$$

where i^* is the ground truth index for the current sample and $|\cdot|_+$ is the ReLU function.

Results: This approach was evaluated on the first task proposed in the benchmark described in Section 4.2.1. This work shown a model that yielded a mAP of 24.10, 39.60 and 49.20 for the 1-shot, 5-shot and 10-shot tasks respectively, outperforming [8] of about 4% of mAP. The authors also introduced a new episodic benchmark based on the `ImageNet-LOC` dataset. As introduced in Section 2.3, in a K-shot N-way episode, N random classes are considered, and for each of them, K random training samples are selected. In this case, 10 query images were

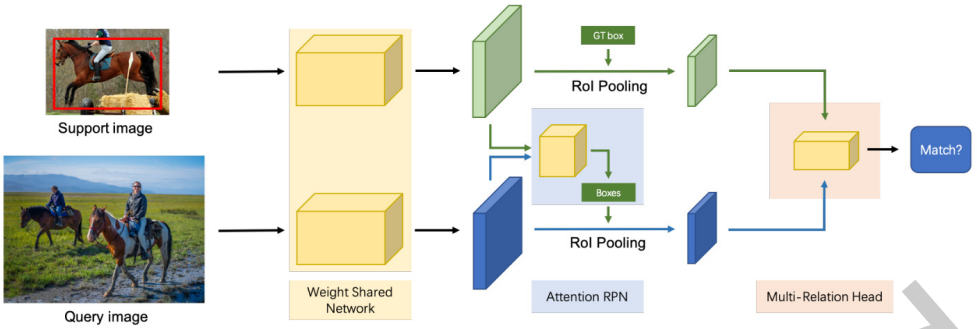

Fig. 10. *Attention-RPN* [13] architecture. The query branch (bottom) consists of a `Faster R-CNN`. The support branch (top), which shares the same backbone of the query, is used to obtain the support feature map. The latter is fed to the Attention RPN along with the query feature map, to predict the RoIs of the query image. A Multi-Relation Head is employed to check the similarity between the proposed boxes and the objects in the support image. Figure from [13].

randomly sampled for each class, resulting in at least 10 instances per class. The model was trained over the first 100 categories from `ImageNet-LOC` and then evaluated over the remaining 214 animal categories unseen at training. The mAP yielded by the model in this setting was respectively 59.20, 73.90, and 79.20 for the three different shots.

4.3.2 Few-Shot Object Detection with Attention-RPN and Multi-Relation Detector (Attention-RPN) [13]. As many DML approaches, this work suggests learning a general matching relationship between the queries and the support set. This enables the detection of novel objects without re-training or fine-tuning. This is achieved by different weight-sharing branches, one for the query set and the others for the support set, which are trained end-to-end with a relation head for the model.

Architecture: As shown in Figure 10, the query branch consists of a `Faster R-CNN` network. Nevertheless, the standard RPN is active in each potential object with a high objectness score, regardless of their actual category; this burdens the subsequent modules with many irrelevant proposals. To address this issue, a novel attention RPN is introduced. It leverages the support information to enable the filtering out of most background boxes and of those in non-matching categories. To this end, the module computes the depth-wise similarity between the support feature map and the query feature map. The result is then used to generate the proposals. Given the support features $\mathbf{X} \in \mathbb{R}^{S\times S\times C}$ and the query features $\mathbf{Y} \in \mathbb{R}^{H\times W\times C}$, where S is the selected kernel size, the similarity is given by the depth-wise cross correlation of the query feature map $\mathbf{Y}$ with support features $\mathbf{X}$ as kernel:

$$\mathbf{G}_{h,w,c} = \sum_{i,j} X_{i,j,c} \cdot Y_{h+i-1,w+j-1,c}, \quad i,j \in \{1,\dots,S\} \tag{6}$$

where $\mathbf{G}$ is the resultant attention feature map. The kernel is computed by averaging on the support feature map. The resulting attention map is given to the *RPN* and its loss $\mathcal{L}_{\text{rpn}}$ is jointly trained with the network. In order to make the detector more discriminative, a multi-relation function is introduced to directly measure the similarity between the proposed boxes in the query and the support objects. The latter employs three attention heads: (i) one that learns global embeddings; (ii) one that learns local correlations between pixels; (iii) and one that learns patch relations. The authors showed that all three are needed for optimal results. Then, they introduced a 2-way contrastive training which consists of randomly selecting: (i) one query image q_c; (ii) one support image s_c containing the same c-th category object; and (iii) another support image $s_{c'}$ containing a different c'-th category

object. Those elements are exploited to construct the training triplet $(q_c, s_c, s_{c'})$, where $c \neq c'$. In this way, the model is able to both match objects from the same categories and distinguishing different categories. In the training triplet, only the c-th category objects in the query image are labeled as foreground, while all other objects are considered as background. During training, the model learns to match every proposal generated by the attention RPN in the query image with the object in the support image. Thus, the model learns to not only match the same category objects between (q_c, s_c), but also distinguish objects in different categories between $(q_c, s_{c'})$. The model is then trained by minimizing $\mathcal{L} = \mathcal{L}_{\text{matching}} + \mathcal{L}_{\text{loc}}$, where $\mathcal{L}_{\text{loc}}$ is the `Fast R-CNN` bounding-box loss and $\mathcal{L}_{\text{matching}}$ is the binary cross-entropy.

Results: The proposed architecture was benchmarked according to the first task proposed in [8] (see Section 4.2.1). With $K = 5$, the model obtained an AP50 score of 44.10, compared with 37.40 achieved by [8]. The model was also evaluated on the `MS-COCO` dataset according to a combination with `VOC` and `ImageNet`, and achieved an AP_{50} of 20.40 and an AP_{75} of 10.60, overcoming [22] and [60] by a significant margin. The authors also introduced a novel dataset for FSOD. Over the latter, already described in Section 3.2, the model yielded an AP_{50} score of 27.50. Finally, they shown that the scores obtained by the model in the `MS-COCO` dataset can be significantly improved by pre-training the model on the new dataset, proving the effectiveness of this specifically designed data collection.

4.3.3 OS2D: One-Stage One-Shot Object Detection by Matching Anchor Features (OS2D). [35] The authors propose a novel model called OS2D, based on a TransformNet especially focused on One-Shot OD, in which the dense grid of anchor locations are taken together from a `Faster R-CNN` and a `SSD`. The proposal consists of a 4-step pipeline: features extraction, correlation matching, spatial alignment, and computing outputs.

Architecture: The proposed architecture is a TransformNet, which is based on a ConvNet. It extracts the features in a Siamese way (i.e., with the same parameters) for both input and class images. Then, a 4D correlation tensor is computed according to the proposal in [42] and then reshaped in 3D to feed a fully connected TrasformNet, to obtain all the transformation parameters for each location of the feature map. These transformations are fed into a grid sampler to produce a spatial alignment where the output consists of a tensor with the coordinates with respect to the input feature map. Finally, the likelihood score of a possible detection in each location is computed based on the grids of points. Localization boxes are then produced by taking maximum and minimum in the grid tensor. Figure 11 graphically describes the entire pipeline.

Results: Two different backbones for the `Faster-R CNN` were tested in the experiments, namely ResNet-50 and ResNet-101. The proposed architecture was firstly trained on the `GroZi-3.2k` dataset [16], with the retrieval system exploiting the weights fine-tuned on `ImageNet` dataset. With this setup, the achieved mAP ranges from 54.50 to 90.60. However, the authors observed that the `GroZi-3.2k` dataset size was not enough to train a network from scratch. To solve this problem, they also compared some pre-trained networks, finally opting for a feature extractor trained on `ImageNet`. To test this setup, the authors exploited the `INSTRE-S1` and `INSTRE-S2` [51] datasets, which contain objects with different orientations. In this context, the architecture achieved a mAP of 88.7 and up to 79.5 with `INSTRE-S1` and `INSTRE-S2`, respectively.

4.3.4 One-Shot Object Detection with Co-Attention and Co-Excitation (CoAE) [20]. Although metric learning is often employed as the key factor for One-Shot OD, its application is not straightforward. The detector needs to know where objects in the target image are more likely to appear to compare them with the query ones by using metric learning. This paper suggests extracting region proposals from non-local feature maps that incorporate co-attention visual cues from query and target images.

Architecture: As depicted in Figure 12, the base architecture is a `Faster R-CNN` with a ResNet-50 as CNN backbone. The first problem arising with this architecture is to examine whether the extracted region proposals are suitable for One-Shot OD. To correctly train the RPN, the authors enriched the convolutional feature maps of interest with the non-local operation [53]. The Squeeze and Co-Excitation (SCE) technique further explores

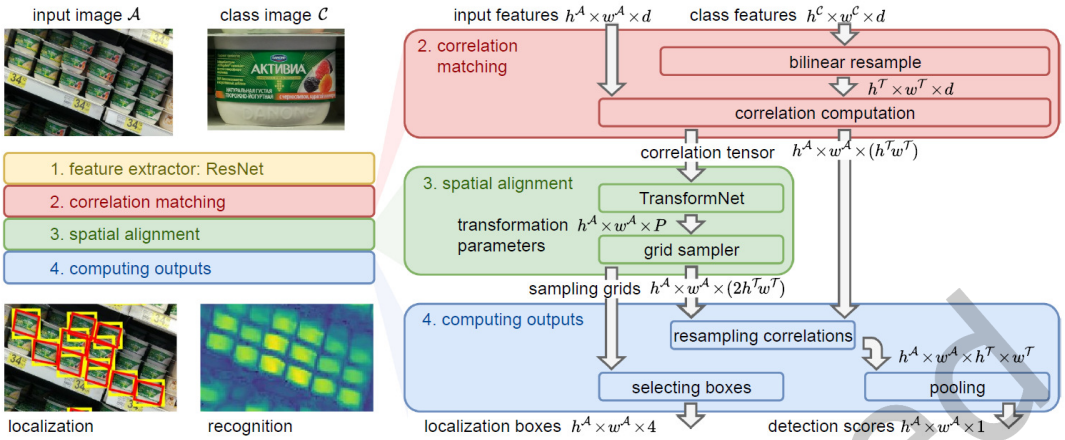

Fig. 11. *OS2D* [35] architecture. Left: the pipeline of data. Right: the details of the three main steps of the OS2D model. Figure from [35].

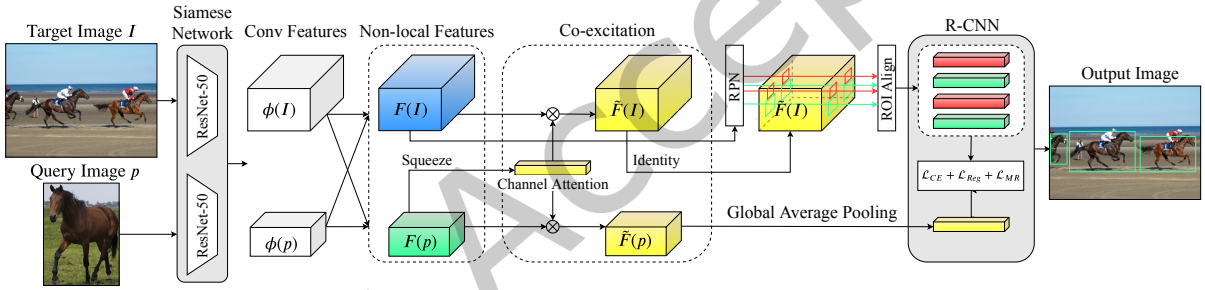

Fig. 12. *CoAE* [20] architecture. Both query image p and target image I are fed to a ResNet-50 to extract features maps. Then, non-local features and the relatedness between the features maps of p and I are computed. Finally, the output containing the bounding boxes of all instances of p in I is provided. Figure from [20].

the relatedness between the feature map extracted from the target image and the feature map extracted from the query patch. In fact, the squeeze step spatially summarizes the feature maps with a global average pooling (GAP), while the co-excitation acts as a bridge between them to emphasize the channels that are crucial in evaluating the similarity measure. During the training, each proposal given by RPN is annotated as foreground or background according to whether their IoU with respect to ground truth is greater than 0.5. A margin-based ranking loss is then considered, ensuring that the most relevant proposals to the query patch would appear in the top part of the ranking list. To compute this loss, a feature vector x is created by combining the vectors r and q, namely the vectors computed from the target image and the query patch after applying SCE and GAP, obtaining $x = [r^T; q^T]^T \in \mathbb{R}^{2N}$. Let $s = \mathcal{M}(x)$ be the foreground probability predicted by the two-layer MLP $\mathcal{M}$,

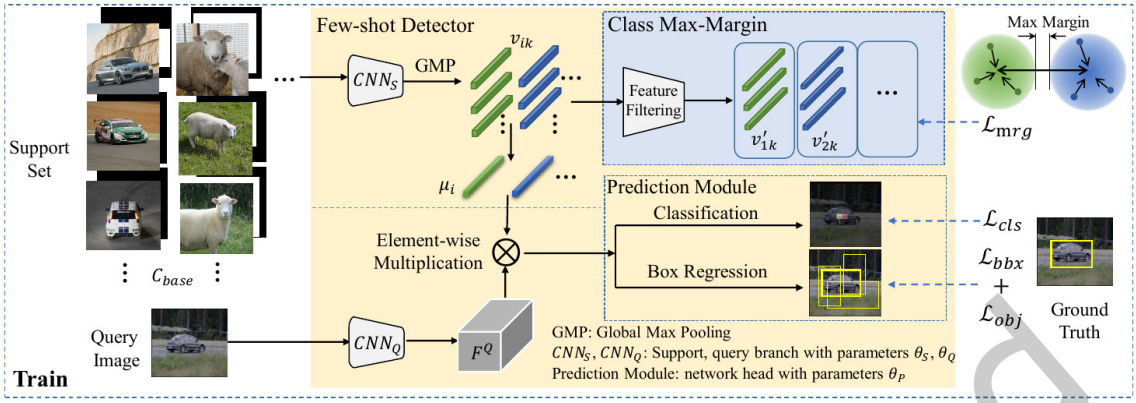

Fig. 13. *CME* [27] architecture. Convolutional feature maps are extracted from support images. Class prototypes are then obtained by averaging them. These are used to reweight the query feature map before the prediction task. Support prototypes are fed to a MLP to filter out localization features and used to compute the max-margin loss. Figure from [27].

whose last layer of the neural network is a two-way softmax. The margin-based ranking loss can be defined as:

$$L_{\text{MR}}(\{x_i\}) = \sum_{i=1}^{K} y_i \times \max\{m^+ - s_i, 0\} + (1 - y_i) \times \max\{s_i - m^-, 0\} + \Delta_i \,, \tag{7}$$

$$\Delta_i = \sum_{j=i+1}^{K} [y_i = y_j] \times \max\{|s_i - s_j| - m^-, 0\} + [y_i \neq y_j] \times \max\{m^+ - |s_i - s_j|, 0\} \,, \tag{8}$$

where $[\cdot]$ is the Iverson bracket, m^+ is the expected lower bound for predicting a foreground proposal, and m^- is the expected upper bound for predicting a background proposal.

Results: The model was evaluated on the `VOC-07/12` benchmark with the addition of the $VOC07_{train}$, and on the `MS-COCO` dataset, using its 2017 version. On the `VOC` dataset, the model pre-trained on all the 1000 classes from `ImageNet` achieved a mAP of 68.20. To assure that the model does not 'foresee' the unseen classes, it was also pre-trained on a reduced training set of `ImageNet`, achieving a mAP of 63.80. On `MS-COCO` dataset, it obtained an AP of 23.60, overcoming the methods chosen for comparison.

4.3.5 Beyond Max-Margin: Class Margin Equilibrium for Few-shot Object Detection (CME) [27]. This work argues about an intrinsic contradiction in the task of FSL in general. In fact, the pair-wise distances among the representations of the base classes should be large to achieve good results in the base class classification. In contrast, their representations should be close enough to obtain low intra-class distance for the novel classes which are represented in terms of features learned from the base classes. To this aim, the authors propose to pursue a *Class Margin Equilibrium* (CME), which allows reaching a trade-off between these two opposing urges.

Architecture: This work exploits the same baseline detector of [22]. As in most *DML* works, the architecture has two parallel branches. In the former, a backbone extracts feature maps from the support images. These are used to obtain prototypes v_{ik} by employing a global max pooling operation. The class prototype μ_i is computed as the mean of the prototypes for that class. These class prototypes are used to highlight or suppress features in the feature maps through an element-wise multiplication. The highlighted features are fed to the prediction module. Differently, in the second branch, query images are fed to a backbone to extract feature maps. The loss is the same as Equation (9). The authors filter out localization features during base training, as these should be

class-agnostic and would therefore perturb class margins. In this regard, they employ a fully connected layer that decouples the localization features to reduce the max-margin loss. Details of the pipeline are shown in Figure 13. To obtain a large margin among the base classes, the prototype vectors must satisfy two conditions: each of them must be both close to its mean prototype and far away from those of different classes. Defining the *intra-class distance* as $D_i^{\text{Intra}} = \sum_{j=0}^{K-1} \left\| v'_{ij} - \mu'_i \right\|_2^2$ and the *inter-class margin distance* as $D_i^{\text{Inter}} = \min_{j, j \neq i} \left\| \mu'_j - \mu'_i \right\|_2^2$, the authors approximate the max-margin by maximizing the upper and lower bounds of margins:

$$\arg\max_{\theta} \mathcal{L}_{mrg} = \frac{\sum_i^N D_i^{\text{Intra}}}{\sum_i^N D_i^{\text{Inter}}}.$$

Another goal is to close this margin when using the features learned from the base classes to reconstruct novel classes during fine-tuning. To this end, they try to reduce the discrimination power of the prototype vectors, also reducing the margin among the classes. This is achieved by employing an online data augmentation procedure that disturbs the features by turning off pixels that correspond to large values in the gradient map of the loss function, as these should have larger discrimination capability. The architecture pursues a class margin equilibrium by maximizing the class margins during back-propagation and minimizing them during the forward pass.

Results: The architecture was evaluated over the `VOC-07/12` and the `MS-COCO` benchmarks. In the former, the authors compared their approach with those based on `YOLO` [8, 22, 55], obtaining better results for $K = 1, 2, 3, 5$. For instance, when $K = 3$ the approach obtained a mAP of 31.5, 27.1 and 30.7 in the three Novel Sets. For $K = 10$, it was outperformed by [55] in two out of three tests. The approach was also compared with those based on the `Faster R-CNN` such as [54, 55, 58–60]. In this case, the approach obtained the best results for $K = 1, 2, 3, 5$ for the second Novel Set, yielding a mAP of 27.2, 30.2, 41.4 and 42.5. In the remaining sets however, [58] often obtained slightly better results. The situation is different in `MS-COCO`, as the approach outperformed [58] both for $K = 10$, achieving an AP of 15.1 and for $K = 30$, obtaining an AP of 16.9. However, the approach was outperformed by [59] when considering different metrics, such as AP50 and AR.

4.3.6 Query Adaptive Few-Shot Object Detection with Heterogeneous Graph Convolutional Networks (QA-FewDet) [19]. This work builds on top of [13], which as described in section 4.3.2 aims at learning a matching relationship between query and support set via two parallel branches and an attention RPN. The authors extend [13] by enriching the pipeline with an auxiliary heterogeneous graph: this is formed by a query-agnostic *inter-class* subgraph and a set of *intra-class* subgraphs which seek to capture relations between classes and classes, proposals and proposals and classes and proposals. Information from the constructed graph is then extracted by employing an ad hoc Graph Convolutional Network and used to empower the final class-proposal matching decision.

Architecture: The approach is based on a `Faster R-CNN` architecture with two Siamese branches, one for the query image and one for support images. As in [13], an `Attention RPN` is used to produce class-specific proposals for each novel class, which are RoI pooled to yield a fixed-size feature map. The class prototype for the novel class is obtained as the average of the feature maps of the corresponding support images. Following [13], the similarity between proposals and class prototypes is computed by a `Multi-Relation Head`. Differently from [13], a heterogeneous graph is built from the proposal and class nodes. As shown in Figure 14, the graph encodes three different types of edges: *class-class*, *class-proposal*, and *proposal-proposal*. The *class-class* edges can be used to enhance novel-class prototypes by exploiting those from relevant classes, while *class-proposal* edges allow class prototypes and feature proposals to mutually adapt. Finally, *proposal-proposal* edges should allow the context of a proposal to be taken into consideration in the prediction. The features generated according to [13] are updated by performing a layer of graph convolutions over the constructed graph. The updated features are used for the pairwise classification.

Results: The approach was evaluated against [13, 22, 54, 55, 58–60] over both `VOC-07/12` and `MS-COCO` benchmarks. In both cases, it outperformed the baselines by a significant margin. In the former, considering $K = 3$, it

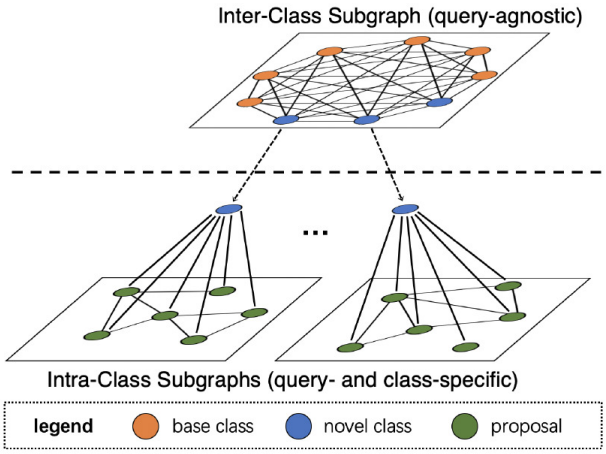

Fig. 14. *QA-FewDet* [19] heterogeneous graph. The graph is composed of two subgraphs: a query-agnostic *inter-class* subgraph and a set of *intra-class* subgraphs. These contain different types of edges to encode relations among proposals and among classes, as well as class-proposal ones. Figure from [19].

obtained an AP50 of 55.7, 46.6 and 47.8 for Novel Sets 1, 2, and 3 respectively. For the latter, which is notably a harder setting where all the approaches achieved far lower results, the proposal obtained an AP of 11.6 and 16.9, for $K = 10$ and $K = 30$ respectively.

4.3.7 Discussion. A common framework can be observed in most of the works in this section. In fact, in [13, 19, 20, 27, 35] the query image and the support image(s) traverse two parallel branches. The support branch is supposed to extract meaningful features from the samples of each class to instill them into the result of the query branch by aggregation. The network should be able to use the enriched features to detect objects of that class in the query image. Moreover, as for many recent works, a crucial role is played by attention, which is used in [13, 19, 20]. Regarding the architecture, `Faster R-CNN` is still the preferred choice, with only [27, 35] employing a one-stage detector. Finally, a meaningful and discriminative embedding space is the key of the success of metric learning approaches: this is achieved in [20, 27, 47] by employing a margin-based loss and in [13] by introducing a contrastive training scheme.

4.4 `Meta-Learning`

As introduced in Section 2.4, `Meta-Learning` is a sub-field of ML focused on models that learn how to learn. In practice, this is implemented by training a meta-learner on a meta-dataset of tasks, each of these having its own train and test datasets, and then evaluating it on another meta-dataset. The meta-learner should be able to extract and exploit meta-knowledge to improve the performance achieved by the learner(s) over the actual downstream tasks. The main difference among the approaches presented in this section is in what is considered as meta-knowledge and the way it is exploited.

4.4.1 Few-Shot Object Detection via Feature Reweighting (MetaYOLO) [22]. A possible way to employ `Meta-Learning` is designing a model that learns how to generalize features among different classes. In this way, when the model faces new classes, it is already trained enough to learn the new weighting coefficients with respect to the general features. The learning of the reweighting coefficients can be performed with few support examples, because the model can exploit the features just for their intrinsic importance in detecting objects belonging to new classes.

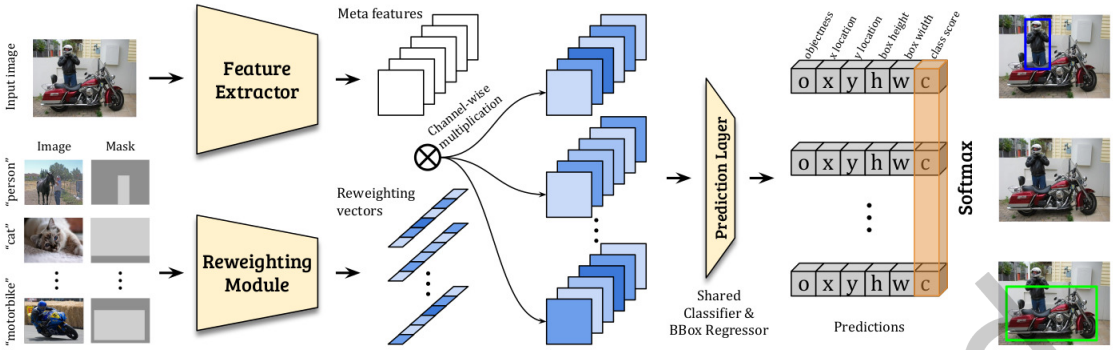

Fig. 15. *MetaYOLO* architecture. *Feature Extractor* extracts meta-features from the input image. *Reweighting Module* obtains reweighting vectors from support images, leveraged to modulate the meta-features via channel-wise multiplication. *Prediction Layer* takes the result of the multiplication and classifies the objects in the input image by regression. Figure from [22].

The aforementioned approach is implemented in [22], where the authors proposed an end-to-end architecture, shown in Figure 15, composed of three modules: (i) A *meta-feature learner* $\mathcal{D}$ that retrieves general, class-agnostic features from the query image; (ii) A *feature reweighting* module $\mathcal{M}$ which extracts discriminative features from the support images in order to assign the weights to the query image meta-features; (iii) A *detection prediction* module $\mathcal{P}$ that predicts classes and bounding boxes for the novel objects.

Architecture: In practice, YOLOv2 is used as the detection prediction module, with its backbone being the meta-feature learner. The feature reweighting module, instead, is designed to be a lightweight *CNN*. Given an input query image I, the meta-features extractor generates meta-features $\mathbf{F} \in \mathbb{R}^{W \times H \times M}$. The reweighting module takes one support image $(\mathbf{X}_i, b_i)$ as input and embeds it into a class-specific representation $\mathbf{w}_i \in \mathbb{R}^m$. This embedding highlights the more relevant meta-features in order to detect the target object from class i. This is achieved by channel-wise multiplying the meta-features $\mathbf{F}$ by the reweighting vector $\mathbf{w}_i$. The authors implemented the reweighting operation through 1×1 depth-wise convolution. The reweighting module is also aware of the target class of the input because it is concatenated to an additional binary mask channel (B_i), which has the value 1 in the positions within the bounding box of the object of interest and 0 in every other place. Class-specific features $\mathbf{F}_i$ are finally fed to the prediction module $\mathcal{P}$, in order to regress the objectness score o, bounding box location offsets (x, y, h, w), and classification score c_i for each of the predefined anchors:

$$\{o_i, x_i, y_i, h_i, w_i, c_i\} = \mathcal{P}\left(\mathbf{F}_i\right), \quad i = 1, \ldots, N$$

where c_i is one-versus-all classification score indicating the probability of the corresponding object to belong to class i. With the purpose of jointly training the model, multiple few-shot detection learning tasks $\mathcal{T}_j$ are created from the training images, with annotations from the base classes. Each task $\mathcal{T}_j = \mathcal{S}_j \cup \mathcal{Q}_j$ is composed of a support set $\mathcal{S}_j$ containing a support image per class for a total of N images, and a query set $\mathcal{Q}_j$ used for the evaluation. The parameters θ_D, θ_M, and θ_P of the meta-feature learner $\mathcal{D}$, reweighting module $\mathcal{M}$ and prediction module $\mathcal{P}$ respectively, are jointly learned by optimizing the following loss:

$$\begin{aligned} \min_{\theta_D, \theta_M, \theta_P} \mathcal{L} := \sum_j \mathcal{L}\left(\mathcal{T}_j\right) &= \sum_j \mathcal{L}_{\text{det}}\left(\mathcal{P}_{\theta_P}\left(\mathcal{D}_{\theta_D}\left(I_q^j\right) \otimes \mathcal{M}_{\theta_M}\left(\mathcal{S}_j\right)\right), M_j^q\right) \\ &= \mathcal{L}_{\text{cls}} + \mathcal{L}_{\text{loc}} + \mathcal{L}_{\text{obj}} \end{aligned} \tag{9}$$

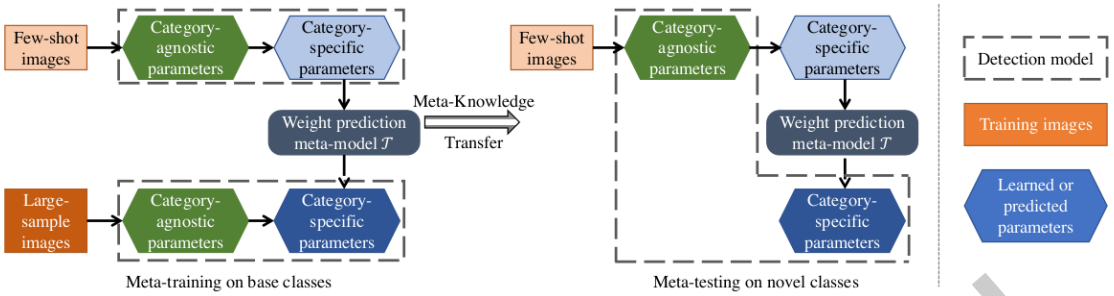

Fig. 16. *MetaDet* [55] architecture. Meta-training: a detector is trained to obtain the category-agnostic parameters. A weight prediction meta-model is trained to convert the category-agnostic parameters into the category-specific ones. Meta-testing: the category-agnostic parameters are initialized from the base detector and the category-specific are predicted leveraging the meta-model. Figure from [55].

where I_q^j and M_j^q represent the query image and its corresponding annotations from the *j-th* task $\mathcal{T}_j$. The authors suggested a learning scheme that evolves into two stages, in order to improve the model generalizability. Those consist in: (i) A base training phase in which the three modules are jointly trained, in order to enforce coordination among them; (ii) A fine-tuning phase in which the model is trained on both the base and the novel classes. Being the second stage a few-shot task, to balance between samples from the base and the novel classes, only K boxes for each base class are used, as this is the number of available boxes for the novel classes. During the whole training phase, the reweighting coefficients depend on the randomly sampled (x, b) pairs; nevertheless, the reweighting vector for a target class is set to the average one predicted by the model after taking the K-shot samples as input. It allows the model to perform detection without requiring any support input. Therefore, the reweighting module can be completely removed during inference.

Results: The authors introduced the 3 different benchmarks which are described in Section 3.3. In the `VOC-07/12` benchmark, the proposed architecture outperformed the compared baseline in [8] by a large margin, especially when there are very few examples, such as $K = 1, 2, 3$. In the last case, in fact, the model yielded a mAP of 26.70, 22.70 and 28.40 for the three different splits, compared with a mAP of 7.40, 3.30 and 9.50 obtained by [8]. Analogous results were reported for the `MS-COCO` benchmark, in which the model obtained a mAP of 12.30 when $K = 10$ and 19.00 when $K = 30$, compared with a mAP of 1.10 and 1.40 achieved by [8]. Finally, in the `MS-COCO→VOC` benchmark the model obtained a mAP of 32.29 with $K = 10$, outperforming [8], which achieved 10.99.

4.4.2 Meta-Learning to Detect Rare Objects (MetaDet) [55]. As it is well-known in the literature, features learned by a convolutional network go from generic to specific as the number of layers progresses. Thus, the low-level layers are class-agnostic and can be transferred from the base to the novel classes, while the top-level layers are class-specific and then usually reinitialized. In this work, the authors proposed a `Meta-Learning` framework that predicts the parameters of category-specific components from few examples through a weight prediction meta-model. This idea relies on the assumption that the way the top-level parameters change during the learning process is shared among many classes. Based on this hypothesis, these dynamics could in principle be learned during the training of a detector on data-abundant classes, and then exploiting them to predict the values assumed by the parameters as if they were trained with sufficient data.

Architecture: The authors built their model, named MetaDet, upon the `Faster R-CNN`, in which the convolutional features and *RPN* are treated as category-agnostic components and, therefore, they share parameters between base

and novel classes. On the contrary, the class-specific components of the network cannot be directly transferred between the base and novel classes. The dynamics pattern describing how the parameters change from the ones trained on a small dataset to those trained on a large dataset can be characterized by a generic category-agnostic transformation. In this regard, the authors introduced the parameterized weight prediction meta-model $\mathcal{T}$. Starting from the few-shot parameters, it learns the transformation needed to learn the bounding box detection parameters of a specific category. This novel component can be implemented as a small fully-connected network, which can then be trained end-to-end with the `Faster R-CNN` through a `Meta-Learning` procedure. More in detail, denoting with: (i) $w_{\text{det}}^{c,*}$ the class-specific weights of the last layer of the detection network learned from the large-sample base dataset S_{base} , and with (ii) w_{det}^{c} the corresponding weights learned from the K-shot episode dataset sampled from S_{base} , the weight prediction meta-model $\mathcal{T}(\cdot)$ regresses from w_{det}^{c} to $w_{\text{det}}^{c,*}$ in the model parameter space: $w_{\text{det}}^{c,*} \approx \mathcal{T}\left(w_{\text{det}}^{c};\phi\right)$, where ϕ are class-agnostic learned parameters. The same $\mathcal{T}(\cdot)$ is applied to any class c. The meta-objective function for each class in an episode is given by the ℓ_2^2-norm of the difference between the predicted weights and the well-trained weights plus the `Faster R-CNN` loss shown in Section 2.1:

$$\left\|\mathcal{T}\left(w_{\text{det}}^{c};\phi\right)-w_{\text{det}}^{c,*}\right\|^2+\lambda\sum_{(x,y)\in \text{RoI}^c}\mathcal{L}\left(\mathcal{D}\left(x;\mathcal{T}\left(w_{\text{det}}^{c};\phi\right)\right),y\right). \tag{10}$$

The final loss is minimized with respect to ϕ, which is averaged over all $c \in C_{\text{base}}$ and over w_{det}^{c} generated in all the episodes. It is worth noticing that the bounding box detection branch contains two types of detection weights: the RoI classification weights w_{cls}^{c} and the bounding box regression weights w_{loc}^{c}. Therefore, these weights are given by the concatenation of the two types of weights: $w_{\text{det}}^{c}=\left[w_{\text{cls}}^{c},w_{\text{loc}}^{c}\right]$. As usual, `Meta-Learning` consists of the two phases illustrated in Figure 16: meta-training on S_{train} and meta-testing on S_{novel} . In this approach, meta-training is furtherly split into two stages for category-agnostic and category-specific components, respectively. During the first stage, a large-sample base detector $\mathcal{D}(\cdot;\theta^*)$ is trained on the entire dataset S_{train} . It produces the base detector that, along with the large-sample parameters $w_{\text{det}}^{c,*}$ of the class specific components, are exploited for the novel classes. The second stage, instead, consists of few-shot episode detection. In each episode, K bounding box annotations per class on S_{train} are randomly sampled. Then, the large-sample base detector $\mathcal{D}(\cdot;\theta^*)$, trained in the first stage, is exploited to generate the K-shot detector. This is achieved by freezing the category-agnostic parameters $\theta^*\backslash w_{\text{det}}^{c,*}$, which were learned in the large-sample setting and only retain the category-specific parameters w_{det}^{c} from scratch. The model is then trained by minimizing Equation (10). Concerning the meta-testing phase, in order to train the K-shot novel detector on S_{novel} , its category-agnostic parameters are initialized from the base detector as $\theta^*\backslash w_{\text{det}}^{c,*}$, while its category-specific parameters w_{det}^{c} are randomly initialized. Thus, the meta-model $\mathcal{T}(\cdot;\phi)$ is used to predict the desired w_{det}^{c} , while fine-tuning the detector. At inference time, the meta-model is detached, and the detector is the standard `Faster R-CNN` one.

Results: The approach was benchmarked over `VOC-07/12` and `MS-COCO`, as described in Section 3.3, and the results were compared with [22]. The former outperformed the latter in both settings: considering `VOC-07/12` with $K = 1$, the proposed model respectively yielded a mAP of 18.90, 21.80 and 20.60 on the 3 different splits, while [22] achieved a mAP of 14.80, 15.70 and 19.20. Over `MS-COCO` benchmark, the mAP obtained by the model was respectively 14.60 and 21.70 for $K = 10$ and $K = 30$, while the mAP yielded by [22] is 12.30 and 19.00. These results confirm the efficacy of an interesting research topic, proving that model weights can be effectively predicted.

4.4.3 Meta R-CNN: Towards General Solver for Instance-level Low-shot Learning (Meta R-CNN)[60]. As described in Section 4.4.2, in [55] the authors proposed a `Meta-Learning` framework based on `Faster R-CNN`. This work exploits the same architecture, but `Meta-Learning` is applied over RoI features instead of on the full image ones. In this context, the authors introduced a *Predictor-head Remodelling Network* (PRN) that shares its backbone with

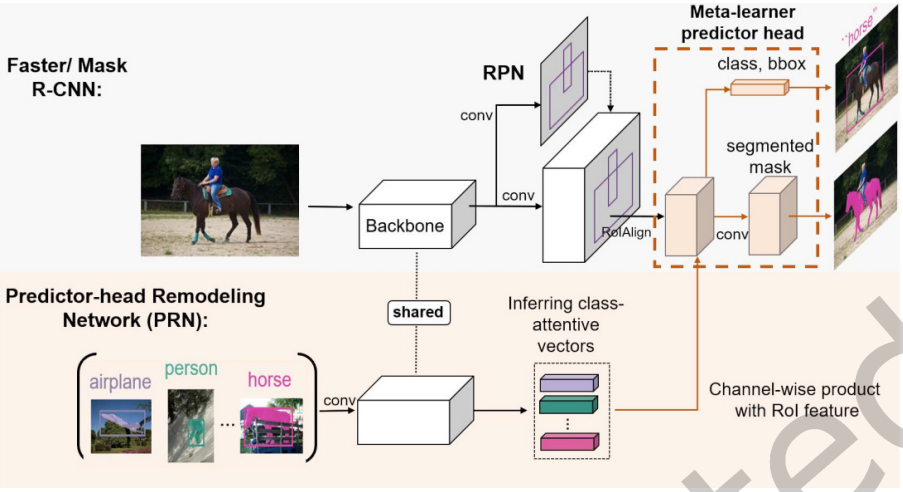

Fig. 17. *Meta R-CNN* [60] architecture. A `Faster R-CNN` (above) and a Predictor-head Remodeling Network (below) are the two main components. The latter shares the backbone of the `Faster R-CNN`, using it to infer the class-attentive vectors from the support set. These vectors are channel-wise multiplied with the RoI features to modulate them. Figure from [60].

the `Faster R-CNN` model. This module aims at meta-learning class-attentive vectors that are used to re-modulate RoI features. The PRN receives images containing objects with their boxes, to infer their class attentive vectors. Such vectors exploit a channel-wise soft-attention on RoI features, remodeling the R-CNN predictor heads to detect or segment the objects that are consistent with the classes these vectors represent.

Architecture: Support images are used in this work to infer object attention vectors $\mathbf{w}^{\text{meta}} = f(S;\boldsymbol{\phi})$ via PRN, in order to build class-attentive vectors by retrieving the average of the object attention vectors for each of the classes: $\mathbf{w}_c^{\text{meta}} = \frac{1}{K}\sum_{j=1}^{K}\mathbf{w}_k^{(c)}$. These vectors can be used to modulate the RoI features $\left\{\hat{\mathbf{z}}_{i,j}\right\}_{j=1}^{n_i}$, in order to detect the objects consistently with the classes they represent. The latter can be achieved by channel-wise multiplying the features and the class-attentive vectors as $\hat{\mathbf{Z}}_i \otimes \mathbf{w}_c^{\text{meta}}$. The attentioned result can then be fed to the standard predictor head in `Faster R-CNN`. An illustration of the architecture is shown in Figure 17. The framework is trained by minimising the following objective function composed of the usual `Faster R-CNN` loss with a *meta-loss* $L(\boldsymbol{\phi})_{\text{meta}}$ (a simple cross-entropy loss), introduced to diversify the inferred object attentive vectors by classifying them in the class each object belongs to, as shown in Equation (11).

$$\min_{\boldsymbol{\theta},\boldsymbol{\phi}} \mathcal{L}(\boldsymbol{\theta},\boldsymbol{\phi})_{\text{cls}} + \mathcal{L}(\boldsymbol{\theta},\boldsymbol{\phi})_{\text{reg}} + \lambda\mathcal{L}(\boldsymbol{\theta},\boldsymbol{\phi})_{\text{mask}} + \mathcal{L}(\boldsymbol{\phi})_{\text{meta}} \tag{11}$$

Results: This approach was evaluated on the same benchmarks of [22], and, therefore, it was used as the baseline. The performance achieved by `Meta-RCNN` on the `VOC-07/12` benchmark with $K = 1$ was a mAP of 19.90, 10.40 and 14.30 for the three different splits respectively, while [22] achieved a mAP of 14.80, 15.70 and 19.20. As it can be noticed, on the last two splits [22] achieved slightly better results. However, increasing the value of K, `Meta-RCNN` outperformed [22], e.g., considering $K = 5$ the mAP yielded by `Meta-RCNN` was 45.70, 34.80 and 41.20, compared with a mAP of 33.90, 30.10 and 40.60 achieved by [22]. On the contrary, considering the more challenging `MS-COCO` benchmark, the mAP achieved by `Meta-RCNN` was respectively 19.10 and 25.30 for $K = 10, 30$ versus a mAP of 12.30 and 19.00 yielded by [22]. The cross-domain generizability of the model was evaluated on the cross-domain benchmark, where `Meta R-CNN` achieved a mAP of 37.40 compared with the 32.30 yielded by [22].

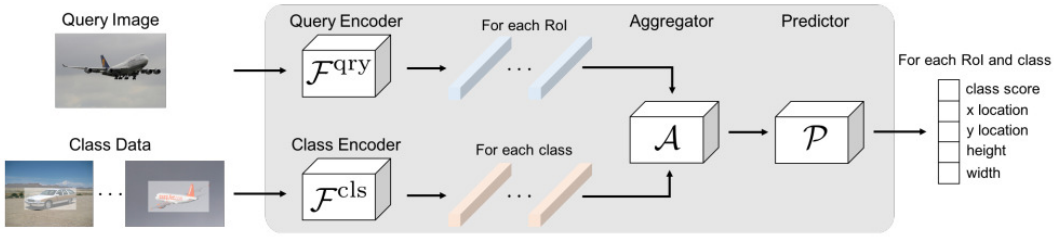

Fig. 18. *FSDetView* [59] architecture. It is based on the `Faster R-CNN`, with the *Query Encoder*, consisting of its backbone, RPN and RoI-align, and the *Class Encoder*, which corresponds to the `Faster R-CNN` backbone. The latter is used to extract the class feature vectors that are aggregated with each RoI feature. Those are provided as input to the *Predictor* for classification and localization. Figure from [59].

4.4.4 Few-Shot Object Detection and Viewpoint Estimation for Objects In The Wild (FSDetView) [59]. Similarly to [60] and [22], this work aims at learning how to extract class-discriminative features from data-abundant base classes to guide the prediction of low-shot novel objects.

Architecture: Support images are fed to the class encoder $\mathcal{F}^{\text{cls}}$, while the query image $\mathbf{X}$ goes through the query encoder. The two branches respectively output the class and the query feature vectors, which are then aggregated by a feature aggregation module $\mathcal{A}$. The result is passed to the predictor head, which estimates a classification score and an object location for each RoI and each class. Similarly to [60], the architecture is based on the `Faster R-CNN`: the query encoder $\mathcal{F}^{\text{qry}}$ consists of its backbone, the RPN and the module that performs the RoI alignment, while the class encoder $\mathcal{F}^{\text{cls}}$ is just the backbone with shared weights. The whole architecture is shown in Figure 18. Loss function and training scheme are the same as [60].

Feature aggregation: Analogously to the proposals in [22] and [60], the authors aggregate the class-representative features with the query features via channel-wise multiplication. In addition, they also include the features obtained as $\mathrm{f}^{\text{qry}} - \mathrm{f}^{\text{cls}}$ and the image embedding f^{qry} untouched, using their channel-wise concatenation as final feature map.

Results: This approach was evaluated on the `VOC-07/12` and the `MS-COCO` benchmarks, outperforming in most cases the `Meta-Learning` approaches presented so far. This is particularly true for $K \geq 3$. In fact, considering the first benchmark, the proposed strategy performed on par or slightly worse than some other models, namely [8], [60], [54], and [22], when $K = 1, 2$. However, it achieved state-of-the-art results for $K = 3, 5, 10$. In particular, in the case of $K = 5$, it reached a mAP of 49.10, 37.00 and 43.80 for the three different splits. In the second benchmark, instead, the model outperformed all the other baselines for each considered values of K, obtaining a mAP of 27.30 and 30.60 for $K = 10$ and $K = 30$ respectively.

4.4.5 Discussion. The approaches in [22, 59, 60] try to obtain meta-knowledge in the form of general features. Thus, when faced with novel classes, the model only has to learn the appropriate reweighting coefficients. These works leverage double-branch architectures similarly to those seen in Section 4.3, both employing two-stage detectors [59, 60] and single ones [22]. On the contrary, [55] proposes an intriguing approach in which a network aims at predicting the fully-trained weights from partially trained ones. This kind of works highlights the flexibility and potential of the meta-learning framework to solve problems in elegant and unconventional ways.

5 ZERO-SHOT LEARNING AND ZERO SHOT OBJECT DETECTION APPROACHES

This section deepens the discussion on the zero-shot clause sketched in Section 2.3.1 providing some more concrete example of proposals dealing with this relatively recent research field. However, the mentioned approaches

are limited to a preliminary analysis due to the fact that the proposed review is focused on FSOD. As already mentioned, the ZSOD [4, 37, 63, 64] comprises all those methods that try to detect objects in a scene without any visual clues about them. Hence, a model has to exploit information that differs from the standard image samples in order to perform such detection. As an example of this concept, attributes such as color, shape, and pose [14, 24, 25], can be used as information source for classifying unseen objects. These types of learning strategies fall in the ZSL category and they are usually subdivided into two subgroups: projection-based and similarity-based methods. The former measure the relation strength between the test samples and the unseen concepts. This operation is accomplished with the projected features of the visual features in the semantic space [2, 24, 43, 65]. Instead, the similarity-based methods [34] determine how close a new instance is to concepts that are already seen by discrete classifiers. In the current state-of-the-art, the most used method to perform ZSL is the combination of images and words. It allows to create descriptions of the images in the training set which can be used for the detection of unseen objects [1, 3]. For example, in [33], the authors proposed a method for both generation and comprehension. In fact, it is able to generate an unambiguous description for objects or regions within an image. While, in [62], the authors proposed a new model to incorporate the context into referring expression models. Even though no data is provided for ZSOD tasks, researchers started investigating the possibility to exploit `Data Augmentation` to create training samples [5, 61]. The first proposes a data-free knowledge transfer based on synthesizing data, effectively providing more teacher behavior for a student to learn. The second proposes a similar method for network quantization, by updating random input to match stored batch norm layer statistics.

6 SUMMARY TABLE AND DISCUSSION

This section provides a discussion about the previously presented works. In particular, Figure 19 shows how these are divided by approach and whether they first generate proposals (two-stage) or directly detect objects (one-stage). `Data Augmentation` is the less represented, with [58] as the only surveyed one. Notice that while the proposal is based on online data augmentation, it is trained in a `Transfer Learning` fashion. More standard `Data Augmentation` pipelines are hard to apply due to the lack of labeled boundary boxes for the augmented images. Works belonging to `Transfer Learning` reach competitive results with simple architectures, which possibly only slightly differ from the abundant data correspondents. `Metric Learning` works enable more complex architectures. In fact, those can be equipped, for example, with different embedding functions for support and query images or contrastive losses to enforce more discriminative embedding spaces. `Meta Learning` allows the most freedom, as a meta-learner can be designed to exploit any kind of meta-knowledge from the source task(s) to solve the downstream task(s). This even encompasses designing models which can directly predict weights for other ones. Concerning the detection framework, two-stage approaches usually obtain better performance at the cost of an increased inference time, as for classical OD. For this reason, most of the surveyed methods are built upon a `Faster R-CNN` detector.

Based on the most relevant information retrieved from the approaches, Table 3 and Table 4 show a summary of results split according to the different metrics adopted. It is also worth pointing out that it is complex to evaluate the most performing solutions among all of the proposed ones. However, the conducted discussion tries to balance the factors to impartially detect strength and weakness points for each work. In general, the mAP metric is used in approaches tested on the various versions of `VOC` dataset. In this context, different values of K are involved, namely 1, 2, 3, 5, 10, and 30. On the contrary, those tested on `MS-COCO` present results in terms of AP, AP50, AP75, APs, APm, APl, AR1, AR10, AR100, ARs, ARm, and ARl. In this case, only a limited number of values of K are evaluated, usually 10 and 30. As a consequence, approaches that are tested on both these datasets are included in the two tables. The methods are also subdivided according to the taxonomy described in Section 1. It

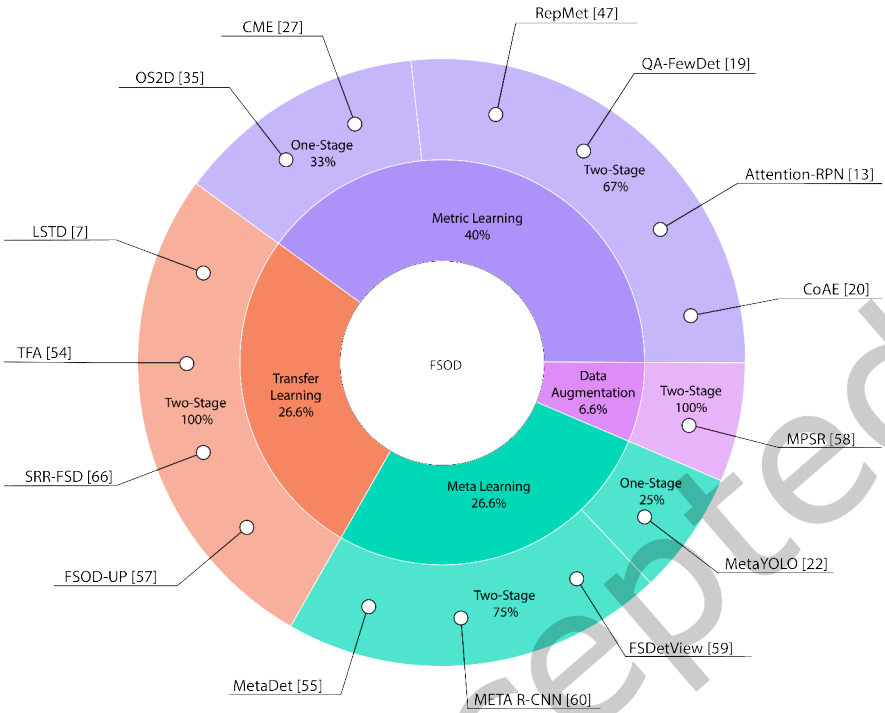

Fig. 19. Categorization of the surveyed works considering approach and type of framework. Most works belong to the `Metric Learning` category, followed by `Transfer Learning` on a par with `Meta Learning`. This division is not strict: [58] is trained in a 2-phase way as in `Transfer Learning` and works as [13, 19, 27] adhere to the `Meta-Learning` framework, while more specifically employing `Metric Learning` techniques.

is critical to underline that could not be possible to directly relate techniques that exploit unequal results metrics and different benchmarks.

As shown in Table 3, the proposals in [58], [54], [66], [57], [27], [19], [22], [55], [60], [59], and [47] (only NS1) rely on the same `VOC-07/12` benchmark. The overall best performance is provided by [19] considering all the K values, excluding $K = 1$. In the latter case, it is overcome by [66] by a significant amount. As a general consideration, the results prove a non-linearity between the achieved performance and the increase of K. In addition, the `Faster R-CNN` is the reference architecture shared by the works that provide the higher scores. In fact, it is generally one of the most used for FSOD, and it seems more robust than others. Differently from the previous table, in Table 4 the values of K presented in the proposals are usually limited to only a few values, e.g. 5, 10, and 30, and the AP and AR scores are provided as a metric. In fact, most of the approaches in this table are tested on the `MS-COCO/VOC` benchmark, namely [58], [54], [66], [57], [27], [19], [22], [55], [60], [59]. The collected results show an overall effectiveness of the approaches based on `DML`, and `Meta-Learning`, e.g., [19] and [59]. In particular, [19] provides the best results in the lower shot cases, i.e., $K = 1, 2, 3, 5$. The overall best performance in terms of AP, AP75, APm and APl are achieved by [27], while [59] seems to be more effective in some of the other metrics. However, there is no clear identification of a specific work or method that overcomes the others, as it happens for the `VOC-07/12` benchmark. Also in this case, the base architecture shared among the best performing approaches is the `Faster R-CNN`. In general, we can highlight the effectiveness of CNNs in this application area. Moreover, the fact that a `Meta-Learning`-based approach turned out to be particularly

Table 3. This summary table includes the proposals for FSOD, testing their approaches with different K values. The mAP is the only evaluation metric used for the analyzed works. The underlined bold results are those achieving the best performance for benchmarks among different works.

Article	Method	Reference Architecture	Dataset								Task	Results					
												K (# shot)					
			VOC07	VOC10	VOC12	MS-COCO	ImageNet 2015	ImageNet-LOC	INSTRE-S1	INSTRE-S2		1	2	3	5	10	30
[58]	DA	FPN Faster R-CNN	Tr/Te	-	Tr	-	-	-	-	-	NS1	41.7	-	51.4	55.2	61.8	-
			Tr/Te	-	Tr	-	-	-	-	-	NS2	24.4	-	39.2	39.9	47.8	-
			Tr/Te	-	Tr	-	-	-	-	-	NS3	35.6	-	42.3	48.0	49.7	-
[8]	TL	SSD Faster R-CNN	-	-	-	Tr	Te	-	-	-	T1	19.2	25.8	-	37.4	44.3	55.8
			Te	-	-	Tr	-	-	-	-	T2	34.0	51.9	-	60.9	65.5	69.7
			-	Te	-	-	Tr	-	-	-	T3	33.6	42.5	-	50.9	54.5	58.3
[54]	TL	FPN Faster R-CNN	Tr/Te	-	Tr	-	-	-	-	-	NS1 FC	36.8	29.1	43.6	55.7	57.0	-
			Tr/Te	-	Tr	-	-	-	-	-	NS2 FC	18.2	29.0	33.4	35.5	39.0	-
			Tr/Te	-	Tr	-	-	-	-	-	NS3 FC	27.7	33.6	42.5	48.7	50.2	-
			Tr/Te	-	Tr	-	-	-	-	-	NS1 CS	39.8	36.1	44.7	55.7	56.0	-
			Tr/Te	-	Tr	-	-	-	-	-	NS2 CS	23.5	26.9	34.1	35.1	39.1	-
			Tr/Te	-	Tr	-	-	-	-	-	NS3 CS	30.8	34.8	42.8	49.5	49.8	-
[66]	TL	Faster R-CNN	Tr/Te	-	Tr	-	-	-	-	-	NS1	**47.8**	50.5	51.3	55.2	56.8	-
			Tr/Te	-	Tr	-	-	-	-	-	NS2	**32.5**	35.3	39.1	40.8	43.8	-
			Tr/Te	-	Tr	-	-	-	-	-	NS3	**40.1**	41.5	44.3	46.9	46.4	-
			Tr/Te	-	Tr	-	-	-	-	-	NS1*	46.3	51.1	52.6	56.2	57.3	-
			Tr/Te	-	Tr	-	-	-	-	-	NS2*	31.0	29.9	34.7	37.3	41.7	-
			Tr/Te	-	Tr	-	-	-	-	-	NS3*	39.2	40.5	39.7	42.2	45.2	-
			Te	-	-	Tr	-	-	-	-	C→V	-	-	-	-	**44.5**	-
[57]	TL	Faster R-CNN	Tr/Te	-	Tr	-	-	-	-	-	NS1	43.8	47.8	50.3	55.4	61.7	-
			Tr/Te	-	Tr	-	-	-	-	-	NS2	31.2	30.5	41.2	42.2	48.3	-
			Tr/Te	-	Tr	-	-	-	-	-	NS3	35.5	39.7	43.9	50.6	**53.5**	-
[47]	DML	RPN	Tr/Te	-	Tr	-	-	-	-	-	NS1	24.1	-	-	39.6	49.2	-
			-	-	-	-	-	Tr/Te	-	-	C1	56.9	-	-	68.8	71.5	-
			-	-	-	-	-	Tr+FT/Te	-	-	C2	59.2	-	-	73.9	79.2	-
[20]	DML	Faster R-CNN	Tr/Val	-	Tr/Val	Tr/Val	-	-	-	-	C3	68.2	-	-	-	-	-
[27]	DML	YOLO	Tr/Te	-	Tr	-	-	-	-	-	NS1	17.8	26.1	31.5	44.8	47.5	-
			Tr/Te	-	Tr	-	-	-	-	-	NS2	12.7	17.4	27.1	33.7	40.0	-
			Tr/Te	-	Tr	-	-	-	-	-	NS3	15.7	27.4	30.7	44.9	48.8	-
		Faster R-CNN	Tr/Te	-	Tr	-	-	-	-	-	NS1	41.5	47.5	50.4	58.2	60.9	-
			Tr/Te	-	Tr	-	-	-	-	-	NS2	27.2	30.2	41.4	42.5	46.8	-
			Tr/Te	-	Tr	-	-	-	-	-	NS3	34.3	39.6	45.1	48.3	51.5	-
[35]	DML	ResNet-50	-	-	-	-	-	-	Tr/Te	-	C4	88.7	-	-	-	-	-
			-	-	-	-	-	-	-	Tr/Te	C4	77.7	-	-	-	-	-
		ResNet-101	-	-	-	-	-	-	Tr/Te	-	C4	88.7	-	-	-	-	-
			-	-	-	-	-	-	-	Tr/Te	C4	79.5	-	-	-	-	-
[19]	DML	Faster R-CNN	Tr/Te	-	Tr	-	-	-	-	-	NS1	42.4	**51.9**	**55.7**	**62.6**	**63.4**	-
			Tr/Te	-	Tr	-	-	-	-	-	NS2	25.9	**37.8**	**46.6**	**48.9**	**51.1**	-
			Tr/Te	-	Tr	-	-	-	-	-	NS3	35.2	**42.9**	**47.8**	**54.8**	**53.5**	-
[22]	MetaL	CNN	Tr/Te	-	Tr	-	-	-	-	-	NS1	14.8	15.5	26.7	33.9	47.2	-
			Tr/Te	-	Tr	-	-	-	-	-	NS2	15.7	15.3	22.7	30.1	40.5	-
			Tr/Te	-	Tr	-	-	-	-	-	NS3	21.3	25.6	28.4	42.8	45.9	-
			Te	-	-	Tr	-	-	-	-	C→V	-	-	-	-	32.3	-
[55]	MetaL	FPN Faster R-CNN	Tr/Te	-	Tr	-	-	-	-	-	NS1	18.9	20.6	30.2	36.8	49.6	-
			Tr/Te	-	Tr	-	-	-	-	-	NS2	21.8	23.1	27.8	31.7	43.0	-
			Tr/Te	-	Tr	-	-	-	-	-	NS3	20.6	23.9	29.4	43.9	44.1	-
[60]	MetaL	PRN Faster R-CNN	Tr/Te	-	Tr	-	-	-	-	-	NS1	19.9	25.5	35.0	45.7	51.5	-
			Tr/Te	-	Tr	-	-	-	-	-	NS2	10.4	19.4	29.6	34.8	45.4	-
			Tr/Te	-	Tr	-	-	-	-	-	NS3	14.3	18.2	27.5	41.2	48.1	-
			Te	-	-	Tr	-	-	-	-	C→V	-	-	-	-	37.4	-
[59]	MetaL	Faster R-CNN	Tr/Te	-	Tr	-	-	-	-	-	NS1	24.2	35.3	42.2	49.1	57.4	-
			Tr/Te	-	Tr	-	-	-	-	-	NS2	21.6	24.6	31.9	37.0	45.7	-
			Tr/Te	-	Tr	-	-	-	-	-	NS3	21.2	30.0	37.2	43.8	49.6	-

Legend of acronyms:

C1=Custom1 | C2=Custom2 | C3= Custom | C4= Custom 4
C→V=MS-COCO→VOC | CS=Cosine Similarity | DA=Data Augmentation | DML=Distance Metric Learning
FC=Fully Connected | FPN=Feature Pyramid Network | FT=Fine Tuning | MetaL=Meta-Learning
NS1=Novel Set 1 (VOC-07/12) | NS2=Novel Set 2 (VOC-07/12) | NS3=Novel Set 3 (VOC-07/12) | PRN=Predictor-head Remodelling Network
RPN=Region Proposal Network | SSD=Single Shot Detection | T1=Task1 | T2=Task2
T3=Task3 | Te=Test Set | TL=Transfer Learning | Tr=Training Set
Val=Validation Set | * Same as VOC-07/12, but novel classes are removed from the pre-training of the backbone network.

Table 4. This summary table includes the proposals for FSOD, testing their approaches with a limited number of different K. Several metrics are provided to enrich the evaluation of the proposed strategy. The underlined bold results are those achieving the best performance for benchmarks among different works.

Article	Method	Reference Architecture	Dataset				Results												
			MS-COCO	LVIS	ImageNet 2015	FSOD	K	AP	AP50	AP75	APs	APm	APl	AR1	AR10	AR100	ARs	ARm	ARl
[58]	DA	FPN Faster R-CNN	Tr/Te *	-	-	-	10	9.8	17.9	9.7	3.3	9.2	16.1	15.7	21.2	21.2	4.6	19.6	34.3
			Tr/Te *	-	-	-	30	14.1	25.4	14.2	4.0	12.9	23.0	17.7	24.2	24.3	5.5	21.0	39.3
[54]	TL	FPN Faster R-CNN	Tr/Te *	-	-	-	FC 10	10.0	-	9.2	-	-	-	-	-	-	-	-	-
			Tr/Te *	-	-	-	FC 30	13.4	-	13.2	-	-	-	-	-	-	-	-	-
			Tr/Te *	-	-	-	CS 10	10.0	-	9.3	-	-	-	-	-	-	-	-	-
			Tr/Te *	-	-	-	CS 30	13.7	-	13.4	-	-	-	-	-	-	-	-	-
			-	Tr/Te	-	-	FC 10	25.4	41.8	27.0	19.8	31.1	39.2	-	-	-	-	-	-
			-	Tr/Te	-	-	CS 10	26.2	41.8	27.5	20.2	32.0	39.9	-	-	-	-	-	-
[66]	TL	Faster R-CNN	Tr/Te *	-	-	-	10	11.3	23.0	9.8	-	-	-	-	-	-	-	-	-
			Tr/Te *	-	-	-	30	14.7	29.2	13.5	-	-	-	-	-	-	-	-	-
[57]	TL	Faster R-CNN	Tr/Te *	-	-	-	10	11.0	-	10.7	4.5	11.2	17.3	-	-	-	-	-	-
			Tr/Te *	-	-	-	30	15.6	-	15.7	**4.7**	15.1	25.1	-	-	-	-	-	-
[13]	DML	Faster R-CNN	Tr	-	FT	Te	5	-	41.3	21.9	-	-	-	-	-	-	-	-	-
			-	-	-	Tr/Te	5	-	41.7	28.3	-	-	-	-	-	-	-	-	-
			-	-	FT	Tr/Te	5	-	44.1	31.0	-	-	-	-	-	-	-	-	-
			Tr+FT	-	-	Te	10	11.1	20.4	10.6	-	-	-	-	-	-	-	-	-
			-	-	-	Tr/Te	10	16.6	31.3	16.1	-	-	-	-	-	-	-	-	-
			-	-	-	Tr/Te	5	-	27.5	19.0	-	-	-	-	-	-	-	-	-
[20]	DML	Faster R-CNN	Tr	-	-	Val	1	-	23.6	-	-	-	-	-	-	-	-	-	-
[27]	DML	YOLO Faster R-CNN	Tr/Te *	-	-	-	10	**15.1**	24.6	**16.4**	**4.6**	**16.6**	**26.0**	16.3	22.6	22.8	6.6	24.7	**39.7**
			Tr/Te *	-	-	-	30	**16.9**	28.0	**17.8**	4.6	**18.0**	**29.2**	17.5	23.8	24.0	6.0	24.6	**42.5**
[19]	DML	Faster R-CNN	Tr/Te *	-	-	-	1	**4.9**	**10.3**	**4.4**	-	-	-	-	-	-	-	-	-
			Tr/Te *	-	-	-	2	**7.6**	**16.1**	**6.2**	-	-	-	-	-	-	-	-	-
			Tr/Te *	-	-	-	3	**8.4**	**18.0**	**7.3**	-	-	-	-	-	-	-	-	-
			Tr/Te *	-	-	-	5	**9.7**	**20.3**	**8.6**	-	-	-	-	-	-	-	-	-
			Tr/Te *	-	-	-	10	11.6	23.9	9.8	-	-	-	-	-	-	-	-	-
			Tr/Te *	-	-	-	30	16.5	**31.9**	15.5	-	-	-	-	-	-	-	-	-
[22]	MetaL	CNN	Tr/Te *	-	-	-	10	5.6	12.3	4.6	0.9	3.5	10.5	10.1	14.3	14.4	1.5	8.4	28.2
			Tr/Te *	-	-	-	30	9.1	19.0	7.6	0.8	4.9	16.8	13.2	17.7	17.8	1.5	10.4	33.5
[55]	MetaL	FPN Faster R-CNN	Tr/Te *	-	-	-	10	7.1	14.6	6.1	1.0	4.1	12.2	11.9	15.1	15.5	1.7	9.7	30.1
			Tr/Te *	-	-	-	30	11.3	21.7	8.1	1.1	6.2	17.3	14.5	18.9	19.2	1.8	11.1	34.4
[60]	MetaL	PRN Faster R-CNN	Tr/Te *	-	-	-	10	8.7	19.1	6.6	2.3	7.7	14.0	12.6	17.8	17.9	**7.8**	15.6	27.2
			Tr/Te *	-	-	-	30	12.4	25.3	10.8	2.8	11.6	19.0	15.0	21.4	21.7	**8.6**	20.0	32.1
[59]	MetaL	Faster R-CNN	Tr/Te *	-	-	-	10	12.5	**27.3**	9.8	2.5	13.8	19.9	**20.0**	**25.5**	**25.7**	7.5	**27.6**	38.9
			Tr/Te *	-	-	-	30	14.7	30.6	12.2	3.2	15.2	23.8	**22.0**	**28.2**	**28.4**	8.3	**30.3**	42.1

* Out of the 80 object classes included in MS-COCO, the 20 classes shared with VOC are exploited as novel classes, while the remaining 60 classes as the base classes.

Legend of acronyms:

AP=Average Precision | AP50=Average Precision 50 | AP75=Average Precision 75 | APl=Average Precision large | APm= Average Precision medium | APs=Average Precision small
AR1=Average Recall 1 | AR10=Average Recall 10 | AR100=Average Recall 100 | ARl=Average Recall large | ARm=Average Recall medium | ARs=Average Recall small
CS=Cosine Similarity | DA=Data Augmentation | FC=Fully Connected | FT=Fine Tuning | MetaL=Meta-Learning | Te=Test Set
TL=Transfer Learning | Tr=Training Set | Val=Validation Set | * MS-COCO benchmark.

valid is an important achievement; indeed, this technique is relatively new and its potentialities are still on an early analysis. As a further general consideration, we can highlight that the MS-COCO/VOC benchmark seems to be challenging, and only few methods [19, 20] try to face it with low values for K, e.g., 1, 2, 3.

The other approaches, instead, must be separately treated. We can notice that the tasks employed in [8], [54], [47], [20], [35], and [13] are different from the standard ones. In fact, in Table 3 NS1, NS2, and N3 are replaced with custom tasks (T1, T2, T3, C1, C2, C3, and C4). This fact implies that the collected results seem tuned due to that. This is particularly true for high values of K, showing good performance. For this reason, a fair direct comparison with those using a common benchmark is not allowed. For the same reason, also in the scenario summarized in Table 4, it is not possible to directly compared these methods with the others. The use of different datasets implies that the achieved performance could provide even higher results than the not-directly comparable works in the overall context. A proof can be provided by [54], tested on LVIS, that seems particularly effective, obtaining very high scores for different AP(s) with $K = 10$.

7 OPEN ISSUES AND FUTURE RESEARCH LINES

This section resumes the main open issues and the possible future research lines for FSOD.

7.1 Open Issues

FSOD suffers from the same issues encountered by OD, worsened by the limited data scenario. In fact, as in OD, low-shot detectors must handle the different appearance variations encountered in the data, such as scales diversity, viewpoint variation, and deformation of non-rigid bodies. These become even more challenging when only a few samples of each class are provided. For instance, the distribution of scales observed in the few supervised samples may be significantly different with respect to the analyzed one. Also the background and foreground indistinguishability may be accentuated in FSOD. In fact, samples from novel and base classes may be localized in different types of backgrounds, hindering the model from detecting the novel objects for which only a few backgrounds are observed. In addition, the scarce data regime may also reduce the robustness against occlusion and cluttered scenes. These challenges are piled up with those from FSL, such as learning overfitted representations from a limited data sample, which leads to low generalizability. This risk is severely increased when facing classes with large intra-class variance and when the small sample is made of non-representative examples. Many realistic use-cases in OD require extremely efficient models to recognize objects in real-time. This need contrasts with accurate predictions, as two-stage detectors are usually more accurate but slower than one-stage ones. This trade-off must be considered in many decisions, such as the images resolution, the type of the backbones, and the dimensionality of the embeddings.

7.2 Future Research Lines

Future directions for FSOD must be sought in both the OD and FSL research scenario. Regarding the former, the field produces continuous evolution in image-processing architectures, which become more expressive and precise in time. These play a crucial role in OD/FSOD, as they are used as backbones to extract rich image features. For instance, transformer-based architectures have proven extremely effective in a series of CV tasks and could replace convolutional architectures in the long run. Other improvements are also likely to come from contaminations of different ML fields, such as NLP and Graph Representation Learning (GRL). In NLP, Self-Supervised Learning (SSL) has proven effective by exploiting the wide amount of unlabeled data to learn general features. Those can be specialized to the task at hand via transfer learning. This strategy has been effectively employed for OD [26] but not for FSOD. Analogously, techniques borrowed from GRL already resulted in advances in OD/FSOD [19]. In fact, the ability of Graph Neural Networks to capture relational patterns has

earned them a role in many successful architectures, with researchers continuously finding new original ways to employ them. For this reason, it is likely that new approaches in FSOD may benefit from their utilization.

Also some FSL methods still have not been adapted to FSOD. Some works introduce an external key-value memory into the pipeline: similarly to metric learning, they aim at learning an embedding function for both the support and query images, with the difference that the embedded query is then compared with the keys of the external memory. Finally, an intriguing research direction consists of letting a network directly generate the weights of another [45]. These have already been effectively used in FSC, but still have not been extended to perform localization.

8 CONCLUSIONS

FSOD is one promising field of CV with high margin of improvement. The ability of an automated system to be able to discern among objects by only seeing a few of samples for each of them is a high desirable characteristics, especially when it is not possible to retrieve a large amount of data. To this aim, different approaches categories have been studied, namely `Data Augmentation`, `Transfer Learning`, `Distance Metric Learning`, and `Meta-Learning`. In the last years, an even more restricted branch of few-shot has been introduced, namely the zero-shot, in which the system is required to discern objects by only having a textual description. It could also be classified as a cross-topic field, between CV and NLP. However, since the core analysis of the review is focused on some of the most relevant proposals in FSOD, both the zero-shot scenario and the domain adaptation topic are only introduced. The conducted study highlighted the effectiveness of some approaches that are well-known in the literature of other application areas. In particular, the use of the `Faster R-CNN`, also in combination with FPN, shows relevant scores in terms of AP and AR. Moreover, the `Meta-Learning` demonstrates as a surprisingly effective method in order to accomplish the task. Another important aspect outlined in the study involves the state-of-the-art benchmarks: currently, the tests for FSOD tasks deal with well-known datasets designed for generic OD. Only one of the analyzed works proposed a dataset for the specific task of FSOD, but it can not be considered as a standard for this field. In general, this application area can be considered still recent. In fact, in the literature, there is only a limited number of works that explore FSOD. Moreover, there is a strong similarity among them in terms of the base architecture exploited in each of the proposed method. The reported results show small variations in terms of scores, underlining this lack of variance. Furthermore, the overall results are still relatively low in terms of precision, showing values below the majority of other OD challenges. We expect that in future works numerous and different approaches will be proposed, in order to try to achieve always more accurate results. Towards this goal, the presented review and taxonomy of techniques on FSOD could aim at fostering progress in research on this topic.

ACKNOWLEDGEMENT

This work was supported by the MIUR grant "Departments of Excellence 2018–2022" of the Sapienza University Computer Science Department and the ERC Starting Grant no. 802554 (SPECGEO).

REFERENCES

[1] Zeynep Akata, Florent Perronnin, Zaïd Harchaoui, and Cordelia Schmid. 2013. Label-Embedding for Attribute-Based Classification. In *2013 IEEE Conference on Computer Vision and Pattern Recognition, Portland, OR, USA, June 23-28, 2013*. 819–826.

[2] Zeynep Akata, Florent Perronnin, Zaid Harchaoui, and Cordelia Schmid. 2016. Label-Embedding for Image Classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38, 7 (2016), 1425–1438.

[3] Zeynep Akata, Scott E. Reed, Daniel Walter, Honglak Lee, and Bernt Schiele. 2015. Evaluation of output embeddings for fine-grained image classification. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*. 2927–2936.

[4] Ankan Bansal, Karan Sikka, Gaurav Sharma, Rama Chellappa, and Ajay Divakaran. 2018. Zero-Shot Object Detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 384–400.

[5] Yaohui Cai, Zhewei Yao, Zhen Dong, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. 2020. Zeroq: A novel zero shot quantization framework. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 13169–13178.
[6] Susan Carey and E. Bartlett. 1978. Acquiring a Single New Word. *Proceedings of the Stanford Child Language Conference* 15 (1978), 17–29.
[7] Hao Chen, Yali Wang, Guoyou Wang, Xiang Bai, and Yu Qiao. 2020. Progressive Object Transfer Detection. *IEEE Transactions on Image Processing* 29 (2020), 986–1000.
[8] Hao Chen, Yali Wang, Guoyou Wang, and Yu Qiao. 2018. LSTD: A Low-Shot Transfer Detector for Object Detection. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*. 2836–2843.
[9] Jiaoyan Chen, Yuxia Geng, Zhuo Chen, Ian Horrocks, Jeff Z Pan, and Huajun Chen. 2021. Knowledge-aware Zero-Shot Learning: Survey and Perspective. *arXiv preprint arXiv:2103.00070* (2021).
[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Fei-Fei Li. 2009. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*. 248–255.
[11] M. Everingham, L. Gool, C. K. Williams, J. Winn, and Andrew Zisserman. 2009. The Pascal Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision* 88 (2009), 303–338.
[12] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. 2012. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.
[13] Qi Fan, Wei Zhuo, Chi-Keung Tang, and Yu-Wing Tai. 2020. Few-Shot Object Detection With Attention-RPN and Multi-Relation Detector. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*. 4012–4021.
[14] Vittorio Ferrari, Manuel J. Marín-Jiménez, and Andrew Zisserman. 2009. Pose search: Retrieving people using their pose. In *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*. 1–8.
[15] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, Vol. 70. 1126–1135.
[16] Marian George and Christian Floerkemeier. 2014. Recognizing Products: A Per-exemplar Multi-label Image Classification Approach. In *Computer Vision – ECCV 2014*, David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars (Eds.). 440–455.
[17] Ross B. Girshick. 2015. Fast R-CNN. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*. 1440–1448.
[18] Agrim Gupta, Piotr Dollár, and Ross B. Girshick. 2019. LVIS: A Dataset for Large Vocabulary Instance Segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*. 5356–5364.
[19] Guangxing Han, Yicheng He, Shiyuan Huang, Jiawei Ma, and Shih-Fu Chang. 2021. Query Adaptive Few-Shot Object Detection With Heterogeneous Graph Convolutional Networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 3263–3272.
[20] Ting-I Hsieh, Yi-Chen Lo, Hwann-Tzong Chen, and Tyng-Luh Liu. 2019. *One-Shot Object Detection with Co-Attention and Co-Excitation*. 1–10.
[21] Licheng Jiao, Fan Zhang, Fang Liu, Shuyuan Yang, Lingling Li, Zhixi Feng, and Rong Qu. 2019. A Survey of Deep Learning-based Object Detection. *CoRR* abs/1907.09408 (2019).
[22] Bingyi Kang, Zhuang Liu, Xin Wang, Fisher Yu, Jiashi Feng, and Trevor Darrell. 2019. Few-Shot Object Detection via Feature Reweighting. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*. 8419–8428.
[23] Brenden M. Lake, Tomer D. Ullman, Joshua B. Tenenbaum, and Samuel J. Gershman. 2017. Building machines that learn and think like people. *Behavioral and Brain Sciences* 40 (2017), e253.
[24] Christoph H. Lampert, Hannes Nickisch, and Stefan Harmeling. 2009. Learning to detect unseen object classes by between-class attribute transfer. In *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*. 951–958.
[25] Christoph H. Lampert, Hannes Nickisch, and Stefan Harmeling. 2014. Attribute-Based Classification for Zero-Shot Visual Object Categorization. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36, 3 (2014), 453–465.
[26] Wonhee Lee, Joonil Na, and Gunhee Kim. 2019. Multi-Task Self-Supervised Object Detection via Recycling of Bounding Box Annotations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.
[27] Bohao Li, Boyu Yang, Chang Liu, Feng Liu, Rongrong Ji, and Qixiang Ye. 2021. Beyond Max-Margin: Class Margin Equilibrium for Few-shot Object Detection. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 7359–7368.
[28] Tsung-Yi Lin, Piotr Dollár, Ross B. Girshick, Kaiming He, Bharath Hariharan, and Serge J. Belongie. 2017. Feature Pyramid Networks for Object Detection. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. 936–944.

[29] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. 2014. Microsoft COCO: Common Objects in Context. In *Computer Vision – ECCV 2014*. 740–755.
[30] Yuan-Pin Lin and Tzyy-Ping Jung. 2017. Improving EEG-Based Emotion Classification Using Conditional Transfer Learning. *Frontiers in human neuroscience* 11 (2017), 334–334.
[31] Li Liu, Wanli Ouyang, Xiaogang Wang, Paul W. Fieguth, Jie Chen, Xinwang Liu, and Matti Pietikäinen. 2018. Deep Learning for Generic Object Detection: A Survey. *CoRR* abs/1809.02165 (2018).
[32] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu, and Alexander C. Berg. 2015. SSD: Single Shot MultiBox Detector. *CoRR* abs/1512.02325 (2015).
[33] Junhua Mao, Jonathan Huang, Alexander Toshev, Oana Camburu, Alan L. Yuille, and Kevin Murphy. 2016. Generation and Comprehension of Unambiguous Object Descriptions. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. 11–20.
[34] Mohammad Norouzi, Tomás Mikolov, Samy Bengio, Yoram Singer, Jonathon Shlens, Andrea Frome, Greg Corrado, and Jeffrey Dean. 2014. Zero-Shot Learning by Convex Combination of Semantic Embeddings. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*.
[35] Anton Osokin, Denis Sumin, and Vasily Lomakin. 2020. OS2D: One-Stage One-Shot Object Detection by Matching Anchor Features. In *Computer Vision – ECCV 2020*, Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm (Eds.). 635–652.
[36] Hang Qi, Matthew Brown, and David G. Lowe. 2017. Learning with Imprinted Weights. *CoRR* abs/1712.07136 (2017).
[37] Shafin Rahman, Salman H. Khan, and Nick Barnes. 2019. Transductive Learning for Zero-Shot Object Detection. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*. 6081–6090.
[38] Sachin Ravi and Hugo Larochelle. 2017. Optimization as a Model for Few-Shot Learning. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.
[39] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. 2016. You Only Look Once: Unified, Real-Time Object Detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. 779–788.
[40] Joseph Redmon and Ali Farhadi. 2017. YOLO9000: Better, Faster, Stronger. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. 6517–6525.
[41] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. 2015. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*. 91–99.
[42] Ignacio Rocco, Relja Arandjelović, and Josef Sivic. 2019. Convolutional Neural Network Architecture for Geometric Matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 41, 11 (2019), 2553–2567.
[43] Bernardino Romera-Paredes and Philip H. S. Torr. 2015. An embarrassingly simple approach to zero-shot learning. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*. 2152–2161.
[44] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. 2015. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)* 115, 3 (2015), 211–252.
[45] Andrei A. Rusu, Dushyant Rao, Jakub Sygnowski, Oriol Vinyals, Razvan Pascanu, Simon Osindero, and Raia Hadsell. 2019. Meta-Learning with Latent Embedding Optimization. In *International Conference on Learning Representations*.
[46] Victor Garcia Satorras and Joan Bruna Estrach. 2018. Few-Shot Learning with Graph Neural Networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*.
[47] Eli Schwartz, Leonid Karlinsky, Joseph Shtok, Sivan Harary, Mattias Marder, Sharathchandra Pankanti, Rogério Schmidt Feris, Abhishek Kumar, Raja Giryes, and Alexander M. Bronstein. 2018. RepMet: Representative-based metric learning for classification and one-shot object detection. *CoRR* abs/1806.04728 (2018).
[48] Jake Snell, Kevin Swersky, and Richard S. Zemel. 2017. Prototypical Networks for Few-shot Learning. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. 4077–4087.
[49] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip H. S. Torr, and Timothy M. Hospedales. 2018. Learning to Compare: Relation Network for Few-Shot Learning. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*. 1199–1208.
[50] Oriol Vinyals, Charles Blundell, Tim Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. 2016. Matching Networks for One Shot Learning. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*. 3630–3638.
[51] Shuang Wang and Shuqiang Jiang. 2015. INSTRE: A New Benchmark for Instance-Level Object Retrieval and Recognition. *ACM Trans. Multimedia Comput. Commun. Appl.* 11, 3 (2015), 21 pages.
[52] Tao Wang, Xiaopeng Zhang, Li Yuan, and Jiashi Feng. 2019. Few-shot adaptive faster r-cnn. (2019), 7173–7182.

[53] X. Wang, R. Girshick, A. Gupta, and K. He. 2018. Non-local Neural Networks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 7794–7803.
[54] Xin Wang, Thomas E. Huang, Joseph Gonzalez, Trevor Darrell, and Fisher Yu. 2020. Frustratingly Simple Few-Shot Object Detection. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*. 9919–9928.
[55] Yu-Xiong Wang, Deva Ramanan, and Martial Hebert. 2019. Meta-Learning to Detect Rare Objects. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*. 9924–9933.
[56] Yaqing Wang, Quanming Yao, James T Kwok, and Lionel M Ni. 2020. Generalizing from a few examples: A survey on few-shot learning. *ACM Computing Surveys (CSUR)* 53, 3 (2020), 1–34.
[57] Aming Wu, Yahong Han, Linchao Zhu, Yi Yang, and Cheng Deng. 2021. Universal-Prototype Augmentation for Few-Shot Object Detection. *CoRR* abs/2103.01077 (2021).
[58] Jiaxi Wu, Songtao Liu, Di Huang, and Yunhong Wang. 2020. Multi-scale positive sample refinement for few-shot object detection. In *European Conference on Computer Vision*. 456–472.
[59] Yang Xiao and Renaud Marlet. 2020. Few-shot object detection and viewpoint estimation for objects in the wild. In *European Conference on Computer Vision*. 192–210.
[60] Xiaopeng Yan, Ziliang Chen, Anni Xu, Xiaoxi Wang, Xiaodan Liang, and Liang Lin. 2019. Meta R-CNN: Towards General Solver for Instance-Level Low-Shot Learning. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*. 9576–9585.
[61] Hongxu Yin, Pavlo Molchanov, Jose M. Alvarez, Zhizhong Li, Arun Mallya, Derek Hoiem, Niraj K. Jha, and Jan Kautz. 2020. Dreaming to Distill: Data-Free Knowledge Transfer via DeepInversion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 1–10.
[62] Licheng Yu, Patrick Poirson, Shan Yang, Alexander C. Berg, and Tamara L. Berg. 2016. Modeling Context in Referring Expressions. In *Computer Vision – ECCV 2016*. 69–85.
[63] Licheng Zhang, Xianzhi Wang, Lina Yao, Lin Wu, and Feng Zheng. 2020. Zero-Shot Object Detection via Learning an Embedding from Semantic Space to Visual Space. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*. 906–912.
[64] Licheng Zhang, Xianzhi Wang, Lina Yao, and Feng Zheng. 2020. Zero-Shot Object Detection with Textual Descriptions Using Convolutional Neural Networks. In *2020 International Joint Conference on Neural Networks (IJCNN)*. 1–6.
[65] Ziming Zhang and Venkatesh Saligrama. 2015. Zero-Shot Learning via Semantic Similarity Embedding. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*. 4166–4174.
[66] Chenchen Zhu, Fangyi Chen, Uzair Ahmed, Zhiqiang Shen, and Marios Savvides. 2021. Semantic Relation Reasoning for Shot-Stable Few-Shot Object Detection. *CoRR* abs/2103.01903 (2021).
[67] Zhengxia Zou, Zhenwei Shi, Yuhong Guo, and Jieping Ye. 2019. Object detection in 20 years: A survey. *arXiv preprint arXiv:1905.05055* (2019).