A Survey of Self-Supervised and Few-Shot Object Detection

Gabriel Huang, Issam Laradji, David Vázquez, Simon Lacoste-Julien, Pau Rodríguez

Abstract—Labeling data is often expensive and time-consuming, especially for tasks such as object detection and instance segmentation, which require dense labeling of the image. While few-shot object detection is about training a model on *novel* (unseen) object classes with little data, it still requires prior training on many labeled examples of *base* (seen) classes. On the other hand, self-supervised methods aim at learning representations from unlabeled data which transfer well to downstream tasks such as object detection. Combining few-shot and self-supervised object detection is a promising research direction. In this survey, we review and characterize the most recent approaches on few-shot and self-supervised object detection. Then, we give our main takeaways and discuss future research directions. Project page: https://gabrielhuang.github.io/fsod-survey/

Index Terms—self-supervised, few-shot, low-data, object detection, instance segmentation

1 Introduction

Traditional object detectors rely on large supervised object detection datasets such as Pascal VOC [28] and MS COCO [71], which have over hundreds and thousands of annotated examples per object category. However, labeling data is often expensive and time-consuming. This is especially true in the case of object detection and instance segmentation, which require dense labeling of bounding boxes/masks for each object, a process that is slower and requires more annotator training than for object classification. Moreover, for fine-grained object detection applications such as plant or animal species recognition, pre-labeled datasets may not exist, and labels may have to be collected on the spot by expert annotators.

To try to solve these problems, few-shot object detection (FSOD) methods attempt to recognize *novel* (unseen) object classes based only on a few examples, after training on many labeled examples of *base* (seen) classes. Until recently, the standard approach in few-shot object detection was to pretrain a backbone for ImageNet classification, then train an object detector on top of this backbone on the base classes, and finally finetune on the novel classes [54, 110, 115, 121, 126] However, because of the tremendous progress in learning self-supervised representations, several (few-shot) detection methods now initialize their backbone from representations pretrained with unsupervised pretext tasks on ImageNet and MS COCO [6, 68, 86, 114, 118, 122].

The problem with typical self-supervised pretrained methods such as SimCLR [15] or MoCo [44] is that they are geared towards classification, and often engineered to maximize *Top-1* performance on ImageNet [111]. However, some of the learned invariances in classification (e.g. to translation) might not be desirable in localization tasks, and thus the representation might discard critical information for object detection. Moreover, it has been shown that higher ImageNet Top-1 accuracy does not necessarily guarantee higher object detection performance [111].

In response to such shortcomings, there has been a growing number of methods for self-supervised object detection. These methods [23, 111, 114, 118, 122] not only attempt to remedy the shortcomings of classification-geared representations, but also pretrain more components in addition to the feature extractor, such as the region proposal network (RPN) and detection head, in the case of Faster R-CNN based methods. In particular, the current state-of-the-art for FSOD on MS COCO is a method which does self-supervised pretraining of both the backbone and the object detector [6].

Thus, this motivates a survey combining the most recent approaches on few-shot and self-supervised object detection, both of which having not been surveyed before (see Section 2). In the following sections, we briefly summarize key object detection concepts (Section 3). Then we review the few-shot object detection task and benchmarks (Section 4) and we discuss the most recent developments in few-shot object detection (Section 4) and self-supervised object detection pretraining (Section 5). We conclude this survey by summing up the main takeaways, future trends, and related tasks (Sections 6 and 7). We provide a taxonomy of popular few-shot and self-supervised object detection methods in Figure 1, on the base of which this survey is structured.

2 Related Surveys

Jiao et al. [51] and Zaidi et al. [124] survey modern deep-learning based object detection methods. They review several state-of-the-art backbones and compare their parameter counts. They present common datasets benchmarks and evaluation metrics. Jiao et al. [51] categorize object detectors into single-stage and two-stage detectors, and report their respective performances in a large table. Zaidi et al. [124] is more recent and discusses backbone choices extensively, including transformer-based backbones such as

- *G. Huang is a PhD student at Mila & DIRO, Université de Montréal. This work was done while he was an intern at Element AI, a ServiceNow company. E-mail: gabriel.huang@umontreal.ca*
- *I. Laradji, D. Vázquez and P. Rodríguez are with Element AI, a ServiceNow company.*
- *S. Lacoste-Julien is an Associate Professor at Mila & DIRO, Université de Montréal and holds a Canada CIFAR AI Chair.*

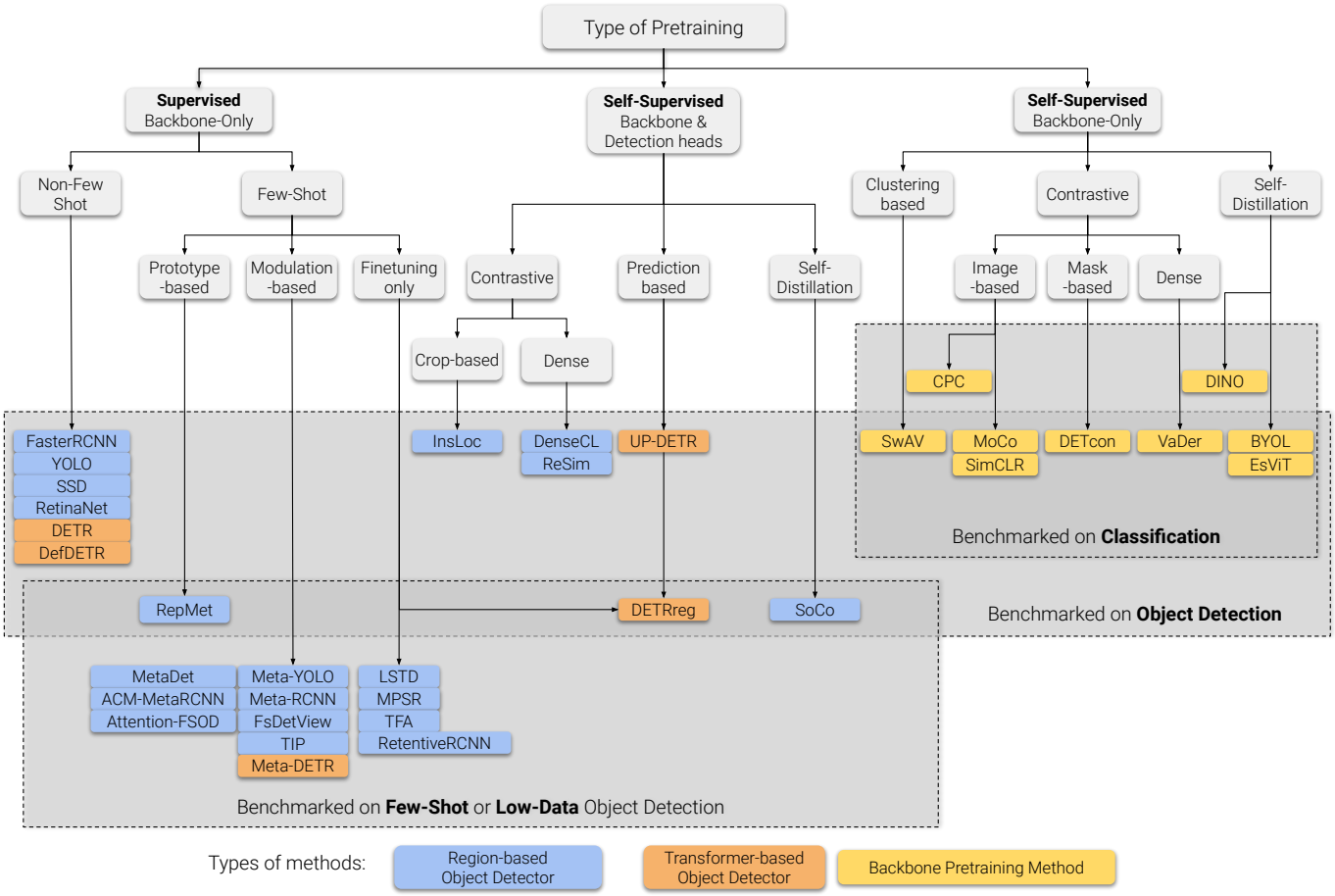

Fig. 1. **A taxonomy of object detection methods reviewed in this survey.** We categorize them based on the following hierarchy: methods using supervised backbone pretraining, methods using self-supervised pretraining of backbone and detection heads, and self-supervised backbone pretraining methods. In parallel, we also tag (shaded rectangles) those methods depending on whether they have been benchmarked on regular object detection, few-shot/low-shot object detection, and ImageNet classification. As discussed in Section 5, many self-supervised classification methods have also been used to initialize object detection backbones and evaluated on object detection benchmarks. DETReg [6], which is a self-supervised object detection method, obtained state-of-the-art FSOD results on MSCOCO and uses self-supervised pretraining of the entire architecture.

Swin transformers [76], and lightweight architectures such as MobileNet [47] and SqueezeNet [50]. Although both these works focus on traditional object detection, they briefly mention weakly-supervised, few-shot and unsupervised object detection as future trends. Note that these surveys do not cover the newer transformer-based object detectors such as DETR [9] or Deformable DETR [132], which we will briefly introduce in this survey.

Jing and Tian [53] present a survey on self-supervised visual feature learning. They perform an extensive review of self-supervised pretext tasks, backbones, and downstream tasks for image and video recognition. In this work, we also introduce a number of generic self-supervised pretraining techniques but we focus on methods particularly designed for object detection.

Regarding few-shot classification, a simpler task than few-shot object detection, Chen et al. [19] introduce a comparative analysis of several representative few-shot classification algorithms. Wang et al. [112] propose a more extensive survey on methods and datasets. However, they do not explore few-shot object detection methods.

Khan et al. [56] show that transformers have achieved impressive results in image classification, object detection, action recognition and segmentation. In this survey, we review object detection methods using transformers as backbones and as detection heads with DETR and variants [9, 132]. We also discuss the emergent properties of visual transformers as showcased by Caron et al. [13].

3 Background on Object Detection

3.1 Key Concepts

For clarity, we start by reviewing key concepts in object detection, and introduce relevant vocabulary. Readers already familiar with object detection can skip directly to Sections 4 and 5 for few-shot and self-supervised object detection. We illustrate those concepts in the context of Faster R-CNN [95] with Feature Pyramid Network [72], a multi-scale two-stage object detector represented in Figure 2, and DETR [9] represented in Figure 3. A more in-depth analysis of object detection concepts can be found in the object detection surveys by Jiao et al. [51], Zaidi et al. [124].

Object detection is the task of jointly localizing and recognizing objects of interest in an image. Specifically, the

object detector has to predict a bounding box around each object, and predict the correct object category. Object detectors are traditionally trained on labeled object detection datasets such as Pascal VOC [28] or MS COCO [71]; the objects of interest are simply the categories that the model is trained to recognize.

The **backbone** network is a feature extractor which takes as input an RGB image and outputs one or several feature maps [72]. Typically, the backbone is a residual network such as the ResNet-50 [42], and is pretrained on ImageNet classification before finetuning it to downstream tasks [6, 110, 121]. Alternatively, an increasing number of works have considered using visual transformers instead [13, 68, 93]. The RGB image is a 3D tensor in $\mathbb{R}^{W \times H \times 3}$, where typically $W = H = 224$ for classification, and $W, H \approx 1300, 800$ for object detection (as per Detectron2's[1] default parameters). For few-shot object detection, a ground-truth mask delimiting the support object is sometimes appended to a fourth channel of the image, and the backbone is modified to take as input tensors in $\mathbb{R}^{W \times H \times 4}$.

Single-scale features consist of a single 3D tensor obtained by taking the outputs of a certain backbone layer. Typically, the C4 layer (corresponding to the output of "res5" the 4th residual block) of the ResNet-50 is used for object detection. The feature map is of size $Z \in \mathbb{R}^{w \times h \times c}$ where c is the number of channels and w, h are the spatial dimensions of the feature map, which are much smaller than the image due to strided convolutions.

Multi-scale features consist of several 3D tensors at different scales. Merely combining several layer outputs from the backbone would result in the high-resolution lower layers having limited semantic information. A common solution is to implement top-down and lateral pathways using a Feature Pyramid Network (**FPN**) [72] to propagate information from the higher-levels of the backbone back to the lower levels (illustrated in Figure 2).

Faster R-CNN [95], represented in Figure 2, is a popular two-stage object detector. To detect objects, they start by feeding an image to the backbone to get single or multi-scale features. Then, they apply the following two stages:

- *Stage 1*: They feed the features to the **Region Proposal Network** (RPN) to extract **object proposals**, which are bounding boxes susceptible to contain objects. The object proposals are predicted at predefined locations, scales and aspect ratios (known as **anchors**), refined using a regression head (anchor deltas), and scored for "**objectness**". They use Non-Maximum Suppression (**NMS**) to remove redundant and low-quality object proposals.
- *Stage 2*: For each object proposal they extract a pooled feature map by resampling the features inside its bounding box to a fixed size, using RoIAlign or ROIPool (pooling strategies). For multiscale-features, the appropriate level is picked using a heuristic. Then, they feed the pooled features into the **Box Head** or **Region-of-Interest** (ROI) head, which predicts the object category and refines the bounding box with another regression head. Finally, they run NMS again to remove redundant and low-confidence predictions.

1. https://github.com/facebookresearch/detectron2

In this survey, we will refer to the union of the RPN and box head as the **detection heads**.

Mask R-CNN [43] is an improvement on top of Fast R-CNN to solve instance segmentation. At the simplest level, Mask R-CNN predicts segmentation masks for each detected instance, additionally to the bounding box and class predictions.

Single-stage object detectors, such as You Only Look Once (**YOLO**) [93, 94], Single-Shot Detector (**SSD**) [75], and newer methods such as RetinaNet [73] and CenterNet [131], are generally simpler and faster than two-stage detectors, at the cost of lower prediction quality. Single-stage detectors directly predict objects at predefined locations from the feature map, with the possibility of subsequently refining the box locations and aspect ratios. Please refer to object detection surveys for an in-depth review [51, 124].

DETR [9] represented in Figure 3, which stands for DEtection TRansformer, is a recent transformer-based architecture for end-to-end object detection. Notably, DETR has a much simpler overall architecture than Faster R-CNN and removes the need for the NMS heuristic –which is non-differentiable– by learning to remove redundant detections. However, being set-based, DETR relies on the Hungarian algorithm [80] for computing the prediction loss, which has been shown to be difficult to optimize [100].

To detect objects, they start by feeding an image to the backbone to get features. Then, they feed the features to the transformer **encoder** to obtain encoded features. Finally, they feed 100 (or any number of) learned "query embeddings" to the transformer **decoder**. The transformer decoder attends to the encoded features and outputs up to 100 predictions, which consist of bounding box locations, object categories and confidence scores. The highest-confidence predictions are returned. There is no need for removing redundant predictions using NMS, as the model learns to jointly make non-redundant predictions thanks to the Hungarian loss. During training, the *Hungarian loss* is computed by finding the optimal matching between detected and ground-truth boxes in terms of box location and predicted class. The loss is minimized using gradient descent. We will also refer to the union of transformer encoder and decoder as the **detection heads**.

Deformable DETR [132] is a commonly used improvement over DETR. Deformable DETR uses multi-scale deformable attention modules, which can attend to a small set of learned locations over multiple feature scales, instead of attending uniformly over a whole single-scale feature map. The authors manage to train their model using 10 times less epochs than DETR [132].

3.2 Datasets and Evaluation Metrics

The most popular datasets for traditional object detection are Pascal VOC [28] and MS COCO [71]. Since they have already been widely discussed in the literature, we refer the reader to previous object detection surveys [51, 124]. Pascal VOC and MS COCO have also been adopted by the few-shot object detection (FSOD) and self-supervised object detection (SSOD) communities. We provide an extensive discussion on their use as few-shot benchmarks in Section 4.3. Please also refer to Section 4.3 for a detailed explanation of the

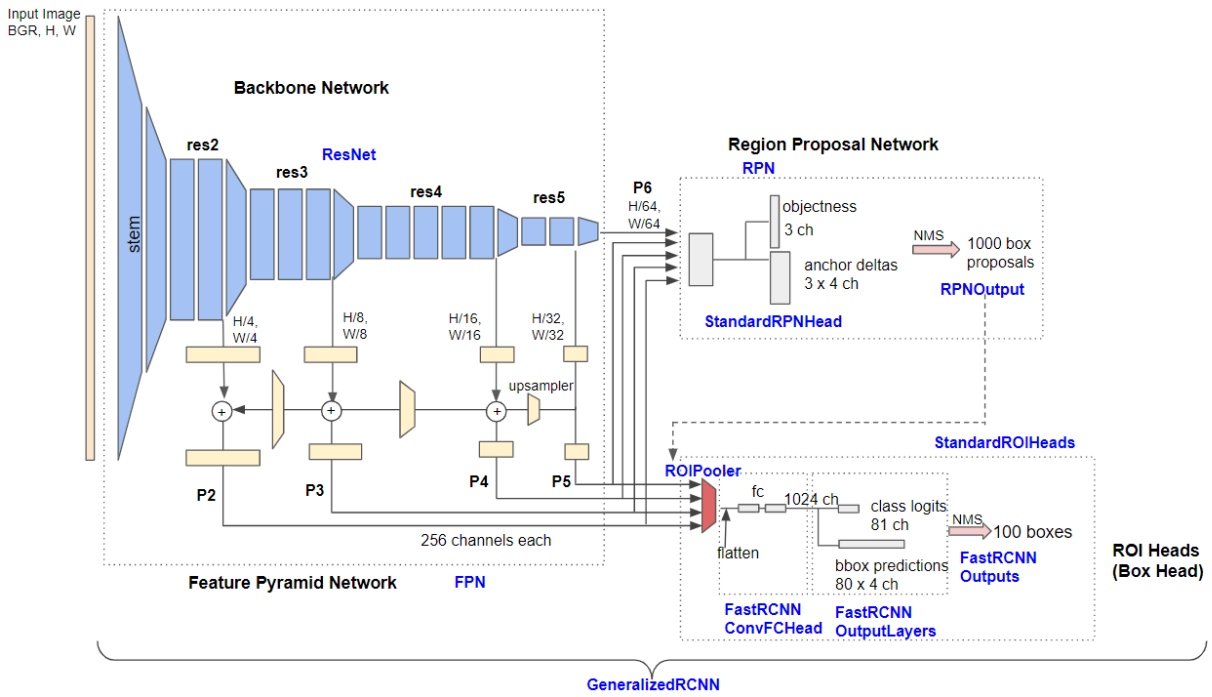

Fig. 2. **A Faster R-CNN with Feature Pyramid Network.** The input image is fed to the backbone network, then the feature pyramid network (light yellow) computes multi-scale features. The region proposal network proposes candidate boxes, which are filtered with non-maximum suppression (NMS). Features for the remaining boxes are pooled with RoIAlign and fed to the box head, which predicts object category and refined box coordinates. Finally, redundant and low-quality predictions are removed with NMS. Blue labels are class names in the detectron2 implementation. Figure courtesy of Hiroto Honda. https://medium.com/@hirotoschwert/digging-into-detectron-2-47b2e794fabd

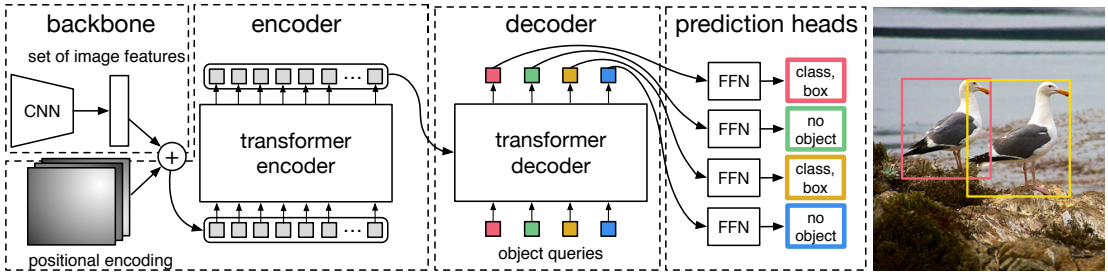

Fig. 3. **The DETR object detector**. The image is fed to the backbone, then positional encodings are added to the features and fed to the transformer encoder. The decoder takes as input object query embeddings, attends to the encoded representation, and outputs a fixed number of object detections, which are finally thresholded, without need for NMS [9]. Image courtesy of Carion et al. [9].

mean average precision (mAP) evaluation metric and the differences between Pascal VOC and MS COCO implementations.

4 Few-Shot Object Detection

Informally, few-shot object detection (FSOD) is the task of learning to detect new categories of objects using only *one* or a *few* training examples per class. In this section, we describe the FSOD framework, its differences with few-shot classification, common datasets, evaluation metrics, and FSOD methods. We provide a taxonomy of popular few-shot and self-supervised object detection methods in Figure 1.

4.1 FSOD Framework

We formally introduce the dominant FSOD framework, as formalized by Kang et al. [54] (**Figure 4**). FSOD partitions objects into two disjoint sets of categories: **base** or known/source classes, which are object categories for which we have access to a large number of training examples; and **novel** or unseen/target classes, for which we have only a few training examples (shots) per class. In the vast majority of the FSOD literature, we assume that the object detector's backbone has already been pretrained on an image classification dataset such as ImageNet (usually a ResNet-50 or 101). Then, the FSOD task is formalized as follows:

(1) **Base training.**[2] Annotations are given only for the base classes, with a large number of training examples per class (*bikes* in the example). We train the FSOD method on the base classes.

2. In the context of self-supervised learning, base-training may also be referred to as *finetuning* or *training*. This should not be confused with *base training* in the meta-learning framework; rather this is similar to the meta-training phase [33].

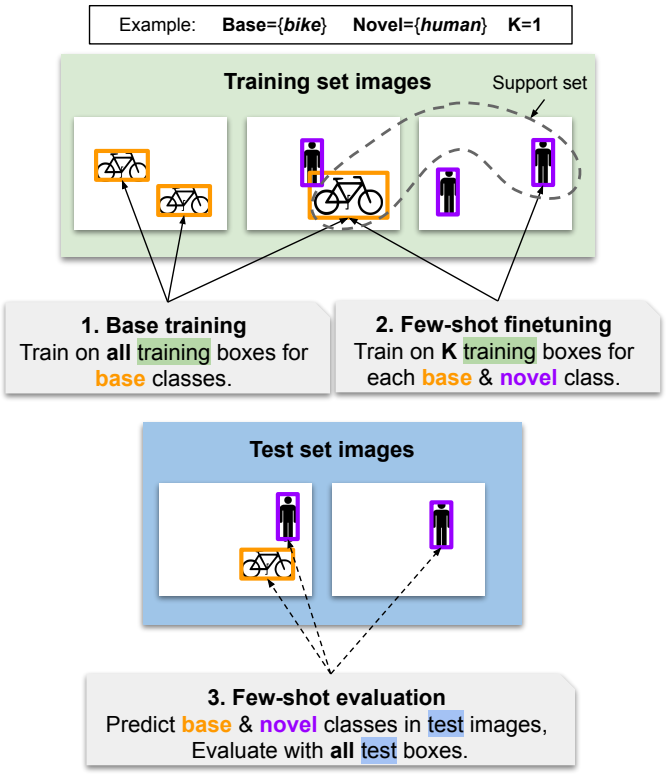

Fig. 4. **Few-shot object detection protocol**, as proposed by Kang et al. [54]. During base-training, the method is trained on base classes. Then during few-shot finetuning, the model is finetuned or conditioned on the support set. Finally, during few-shot evaluation, the method is evaluated on base and novel class detection.

(2) **Few-shot finetuning.** Annotations are given for the *support set*, a very small number of training examples from *both* the base and novel classes (one *bike* and one *human* in the example). Most methods finetune the FSOD model on the support set, but some methods might only use the support set for conditioning during evaluation (finetuning-free methods).

(3) **Few-shot evaluation.** We evaluate the FSOD to jointly detect base and novel classes from the test set (few-shot refers to the size of the support set). The performance metrics are reported separately for base and novel classes. Common evaluation metrics are variants of the mean average precision: mAP50 for Pascal and COCO-style mAP for COCO. They are often denoted bAP50, bAP75, bAP (resp. nAP50, nAP75, nAP) for the base and novel classes respectively, where the number is the IoU-threshold in percentage (see Section 4.4 for full explanation).

In pure FSOD, methods are usually compared solely on the basis of novel class performance, whereas in Generalized FSOD, methods are compared on both base and novel class performances [31]. Note that "training" and "test" set refer to the splits used in traditional object detection. Base and novel classes are typically present in both the training and testing sets; however, the novel class annotations are filtered out from the training set during base training; during few-shot finetuning, the support set is typically taken to be a (fixed) subset of the training set; during few-shot evaluation, all of the test set is used to reduce uncertainty [54].

For conditioning-based methods with no finetuning, few-shot finetuning and few-shot evaluation are merged into a single step; the novel examples are used as support examples to condition the model, and predictions are made directly on the test set. In practice, the majority of conditioning-based methods reviewed in this survey do benefit from some form of finetuning.

4.2 Important Differences with Few-Shot Classification.

As this may not be obvious to readers unfamiliar with both fields, we explicit several practical differences between the FSOD finetuning-based paradigm and the learning-to-learn paradigm [19, 49, 92, 108] commonly used in few-shot classification (FSC):

- **Several objects per image.** In FSOD there can be several instances from base and novel classes in the same image, whereas FSC assumes only one dominant object per image. While FSC filters out all novel *images* during base-training, FSOD removes only the novel object annotations but keeps the images that also contain base objects.
- **Joint prediction on base and novel.** During few-shot finetuning and evaluation, both base and novel classes are present and have to be jointly detected. On the contrary, few-shot classifiers are typically only finetuned and evaluated on novel classes. Note however that many papers only report average precision for novel classes under metric names such as *nAP* or *nAP50*.
- **Learning-to-learn vs. finetuning.** Gradient-based few-shot classification methods such as MAML [33] or Learning-to-Optimize [92] rely heavily on the *learning-to-learn* paradigm; during (meta)training, N-way K-shot episodes are generated by sampling a small training set (support set) and a small validation set (query set) from the base classes. The classifier is finetuned with gradient descent on the support set, makes predictions on the query set, and the query-set loss is minimized using gradient descent, which propagates the gradient through support set tuning. On the contrary, a majority of FSOD methods do not generally consider episodes or backpropagate through gradient descent. Pure finetuning FSOD methods [6, 14, 110, 115] are first trained on all base classes during base training, and finetuned only once on a fixed support set before few-shot evaluation. Moreover, because the few-shot finetuning step (e.g. optimizer learning rates) and the pre-finetuning weights are not calibrated over several episodes using learning-to-learn on a separate query set, they might not be optimal. This is partially mitigated by hyperparameter tuning, which can help find optimal learning rates, but not find the optimal pre-finetuning weights.
- **Episodic Evaluation.** For FSC evaluation, several episodes are sampled from the novel classes [84, 96, 108]; the classifier is finetuned on the support set and classifies the query set, and the results are averaged over hundreds of runs, which have the advantage of reducing variance and estimating confidence intervals [19, 49]. On the contrary, each of Kang's splits [121]

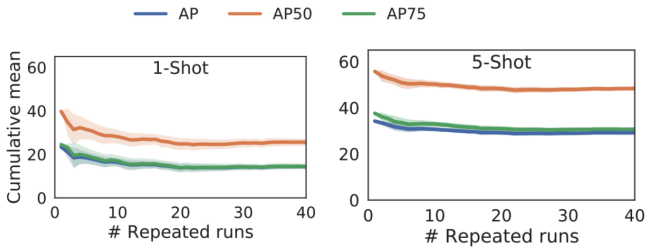

Fig. 5. **Importance of evaluating over several episodes.** The nAP, nAP50 and nAP75 of PASCAL VOC Split-1 are averaged using a variable number of episodes. Note how the means and variances only become stable after around 20 episodes. Figure courtesy of [110].

feature only *one* fixed support set (the exact instances are prespecified), which is known to cause overfitting and overestimating performance especially in the case of 1-shot object detection, when the support set is tiny [110]. See Figure 5 for the impact of using several episodes on Pascal VOC. In response to those variance issues, Wang et al. [110] have proposed new splits for Pascal in which they pre-generate 30 seeds (instead of one) with 30 associated support sets, in an attempt to alleviate this problem.

- **Separate validation and test sets.** Whereas FSC methods generally validate and test on separate splits for common benchmarks such as Omniglot [59], *mini*ImageNet [108], or Synbols [58], FSOD detection methods follow the standard practice in object detection of training on the union of the training and validation sets, and using the test set for both hyperparameter tuning and evaluation, which inevitably leads to overestimating the generalization ability of the methods.

4.3 FSOD Datasets

We describe the dominant FSOD benchmarks, as introduced by Meta-YOLO [54] and improved by TFA (Two-stage Finetuning Approach) [110] to mitigate variance issues. These are also the benchmarks that we will use to compare FSOD methods in Table 3. We compute data statistics in **Table 1**.[3] Caveats and future best practices are discussed in Section 4.3.4 and Section 6.5.

4.3.1 Pascal VOC [28]

Pascal VOC is arguably one of the most popular smaller benchmarks for traditional object detection. For few-shot object detection, the object categories are split between **15 base classes** and **5 novel classes**. Three different base/novel splits are usually considered, denoted splits 1, 2 and 3. For base training, the training and validation (*trainval*) images from VOC2007 and VOC2012 are used, which might lead to overfitting (Section 6.5). This amounts to **40k base boxes** spread over 15k images for an average of **2700 boxes / base class**, with 2.8 boxes / image. For few-shot finetuning, a fixed subset of the VOC2007 and VOC2012 *trainval* sets is taken as the support set. Kang et al. [54] consider the **1,2,3,5,10**-shot settings, which correspond to 15-150 base bounding boxes and 5-50 novel bounding boxes. The fact that the instances are fixed may lead to overestimating performance (Section 4.3.4). For few-shot evaluation, roughly 5k images from the VOC2007 test set are considered, with 13k base and 2k novel boxes, which adds up to an average of **870 boxes/base** and **400 boxes/novel** class. The method has to detect both base and novel classes, and several evaluation metrics are reported separately for base and novel classes: mAP50, mAP75, and COCO-style mAP. The main comparison metric is the novel mAP50.

4.3.2 MS COCO [71]

MS COCO or Microsoft Common Objects in COntext [71], is another popular benchmark for object detection. For FSOD, the object categories are split between **20 novel classes** which are shared with Pascal VOC, and the remaining **60 base classes**. Following Kang et al. [54] and Wang et al. [110], a 5k subset of the COCO2014 validation images are used for few-shot evaluation (denoted *val5k*), while the remaining COCO2014 training and validation images are used for base-training and few-shot finetuning (denoted *trainvalno5k*). Specifically for base training, roughly **367k base boxes** are considered (10 times more than Pascal VOC), spread over 98k images from *trainvalno5k*. This is an average of **6k boxes / base class** and 3.7 boxes/image. For few-shot finetuning, a fixed subset of *trainvalno5k* is taken as the support set. Kang et al. [54] consider the **10,30**-shot settings, which correspond to 600-1800 base boxes and 200-600 novel boxes, and suffer less from overfitting than Pascal VOC [110] despite also using fixed instances. For few-shot evaluation, the *val5k* are used, consisting in 15k base boxes and 20k novel boxes, which amounts to **250 boxes / base** class and **1k boxes / novel** class. Several evaluation metrics are reported separately for base and novel categories: COCO-style mAP, mAP50, mAP75, and mAP for small, medium and large objects. The main comparison metric is novel mAP.

4.3.3 LVIS [41]

LVIS or Large Vocabulary Instance Segmentation [41] is a newer object detection dataset featuring 1230 classes, categorized as *frequent*, *common* (10 or more annotations) and *rare* (less than 10). TFA [110] have proposed using the *v0.5* version of this dataset for FSOD, dividing it into **776 base classes** (frequent and common) and **454 novel classes**, making it by far the FSOD benchmark with the most number of categories (10 times more categories than COCO, 50 times more than Pascal). For this description, we follow the reference implementation from TFA[4] as we could not find all the details in the paper. For base training, **688k base boxes** are considered, spread over 69k images from the training set, which amounts to **887 boxes / base class** and 10 boxes / image. For few-shot finetuning, up to 10 shots from the training set are considered, depending on the number of available examples. This corresponds to an average of 8.57 boxes/class, spread over 7.7k base boxes and 2.8k novel boxes. For few-shot evaluation, the validation set of LVIS is used, consisting of 50,334 base and 429 novel boxes, spread over 5,000 evaluations images. Evaluation

3. Note that the reported number of images is after removing those containing no relevant annotations (e.g. for base training, the images which contained only novel objects are removed).

4. https://github.com/ucbdrive/few-shot-object-detection

TABLE 1
Common FSOD benchmarks. Image counts are after filtering out the ones containing no relevant bounding boxes.

Benchmark	Classes		1. Base Training		2. Few-shot Finetuning			3. Few-shot Evaluation		
	Base	Novel	#images	#**base**-bb	#shots	#**base**-bb	#**novel**-bb	#images	#**base**-bb	#**novel**-bb
Pascal VOC/split 1	15	5	14,631	41,084	1:2:3:5:10	15–150	5–50	4,952	13,052	1,924
Pascal VOC/split 2	15	5	14,779	40,397	1:2:3:5:10	15–150	5–50	4,952	12,888	2,088
Pascal VOC/split 3	15	5	14,318	40,511	1:2:3:5:10	15–150	5–50	4,952	13,137	1,839
MS COCO 2014	60	20	98,459	367,702	10:30	600-1,800	200-600	5,000	15,318	20,193
LVIS v0.5	776	454	68,568	688,029	8.57 (variable)	7,760	2,786	5,000	50,334	429

metrics are COCO-style mAP, mAP50 and mAP75, reported separately for frequent, common, rare objects, and also aggregated over all three categories.

4.3.4 *Discussion*

We discuss some of the advantages and issues associated with the aforementioned benchmarks (see also Section 6.5). For Pascal VOC, the support set is very small (20-200 examples) and the specific instances are predefined by Kang's splits. This can cause overfitting and result in overestimating the novel average precision, especially in the 1-shot case, an issue illustrated by TFA [110] in Figure 5. To mitigate this issue, TFA [110] propose to sample 30 support sets using different random seeds, and to average the results in a benchmark which we will denote the **TFA-splits** as opposed to the benchmark using a single fixed support set, which we will denote the **Kang-splits**. This is a good first step, but not as reliable as the common practice in few-shot classification of averaging metrics over 600 episodes [19].

With a substantial support set of 800-2400 bounding boxes for few-shot finetuning, and plenty of few-shot evaluation boxes for base and novel categories, MSCOCO is arguably the most reliable benchmark for comparing different methods. In fact, we sort methods according to 30-shot MSCOCO nAP in Table 3. Because the 20 novel classes were chosen to in common with Pascal, a very natural benchmark to consider is the MSCOCO⟶Pascal cross-domain scenario, which has been considered by some of the earlier and subsequent works [14, 30, 115].

With 1230 classes, LVIS has more than an order of magnitude more object categories than MSCOCO, and a lot of potential for few-shot object detection. However there are some shortcomings with directly using TFA splits. One problem is that only 705 out of 776 base classes and 125 out of 454 novel classes appear in the validation set, which means that the majority of novel classes will never be evaluated on[5] Moreover, because there are almost 100 more times base than novel boxes in the validation set, performance is completely dominated by base objects for metrics aggregated on base and novel classes. Finally, even the 125 novel classes that are evaluated on only have an average of 3.4 boxes / class, which means evaluation is potentially very noisy. Due to those issues, we do not recommend the current TFA splits for evaluating few-shot object detection methods. However, LVIS has a lot of potential in FSOD given the large diversity of objects. Therefore, proposing more balanced splits and evaluation sets would definitely be beneficial to the FSOD community.

5. As found by running TFA's data loader: https://github.com/ucbdrive/few-shot-object-detection

4.4 FSOD Evaluation Metrics

By design, object detectors such as Faster R-CNN and DETR output a *fixed* number of predictions paired with a confidence score. This means they use a threshold to cut off low-confidence predictions. Therefore, they have to trade off between using a higher threshold, which will lead to higher precision (most predicted objects are actually real objects) but low recall (miss out many of the real objects); and using a lower threshold, which could increase recall at the expense of precision.

The **mean average precision (mAP)** is a standard metric for evaluating object detection and FSOD methods, and is defined as the mean of the individual average precisions (AP) for each object class. The individual APs are defined as the area under the precision-recall curve – discussed below – which can be plotted by varying the confidence threshold of the object detector.

To compute the AP for a class, we first rank the detections for this class by decreasing confidence. Starting from the top-ranked detections ($k = 1$), we consider them as *True Positives* if their intersection over union (IoU) with any ground-truth true object is above a given IoU-threshold (typically 50% or 75%). If the IoU is below the threshold or the ground-truth has already been detected, then we consider them to be *False Negatives*. For each rank k, corresponding to a different choice of confidence-threshold, we can compute the precision@k, a measure of relevance defined as the number of true positives among the top-k boxes divided by k, and the recall@k, a measure of sensitivity defined as the number of true positives among the top-k boxes divided by the total number of ground truth boxes.

We give an example of AP computation in Table 2. Notice how recall is non-decreasing as a function of k, while precision can fluctuate up and down. By varying k between 1 and the total number of detections, we can plot a precision vs. recall curve (see Figure 6). The precision-recall curve (orange) is made non-increasing by taking the *interpolated* precision (green), defined as $p_{interp}(r) = \max_{r' \geq r} p(r)$.[6] The average precision is defined as the area under that curve. The exact way the area is computed depends on the specific benchmark.

6. Interpolation is a natural thing to do because it means that for a minimum recall requirement, there exists a confidence-threshold which results in a detector with better precision if we allow the recall to be higher than that threshold, which is never a detrimental.

TABLE 2
Example computation of precision@k and recall@k. There are 10 detections ranked by decreasing confidence and 5 ground-truth boxes. Example from https://jonathan-hui.medium.com/map-mean-average-precision-for-object-detection-45c121a31173

Rank k	True Positive?	Precision@k	Recall@k
1	True	1.0	0.2
2	True	1.0	0.4
3	False	0.67	0.4
4	False	0.5	0.4
5	False	0.4	0.4
6	True	0.5	0.6
7	True	0.57	0.8
8	False	0.5	0.8
9	False	0.44	0.8
10	True	0.5	1.0

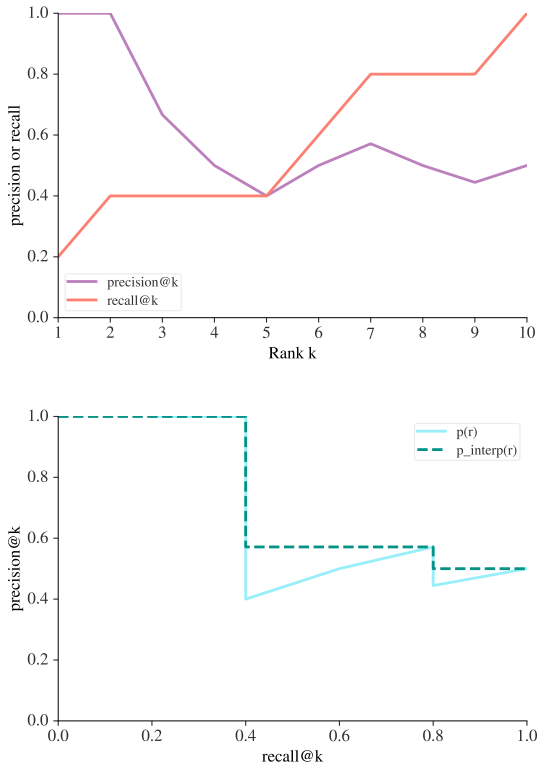

Fig. 6. **Top**: precision@k and recall@k as a function of k the number of boxes considered. **Bottom**: precision@k and interpolated-precision@k as functions of the recall@k (precision-recall curve)

For Pascal VOC's 2007 test set [28], which is used for FSOD evaluation, the main IoU-threshold used is 0.5 and the area under the curve is approximated by sampling and averaging the interpolated precision at 11 points $\{0, 0.1, 0.2, \ldots, 1\}$. IoU-thresholds of 0.75 and COCO-style mAP are also sometimes reported [110]. For MS COCO, the area under the interpolated precision curve is computed exactly by adding up the area of each rectangle (see Figure). COCO-style mAP is defined by averaging the mAP at different thresholds from 0.5 to 0.95 in increments of 0.05. For COCO, it is common to report mAP scores for objects of different sizes: small, medium and large [110, 119]. Additionally, mAP50 and mAP75 are often provided [29, 110, 126], which are computed by computing the exact area under the curve for IoU-thresholds of 0.5 and 0.75. Generally in FSOD, it is also common to report mAP separately for base and novel classes, denoted as bAP and nAP [110]. Most works put greater emphasis on the nAP [54, 121], though some claim that maximizing bAP is also important to avoid catastrophic forgetting [31, 110]. These metrics and datasets give us a comprehensive overview of how different models perform for few-shot object detection. Note that for LVIS v0.5, the metrics are the same as for MS COCO, and reported separately for frequent, common and rare objects. However, we do not recommend following the TFA splits for LVIS due to the issues outlined in Section 4.3.4.

4.5 Few-Shot Object Detection Methods

We review several FSOD methods from the literature. In our description, we assume that backbones have already been pretrained on ImageNet. We summarize most of these methods in Table 3.[7]

4.5.1 *Finetuning-only baselines* [8]

Finetuning-only methods generally start from a traditional object detector such as Faster-RCNN [95] with only minor architecture modifications. They do base training on many base class examples, then do few-shot finetuning on a support set containing both base and novel classes. The rationale behind this two-step process is to deal with the extreme imbalance between the base and novel classes, and to avoid overfitting on the novel classes.

LSTD [14] proposed the first finetuning strategy for FSOD. A hybrid SSD/Faster-RCNN "source" network is trained on the source classes, then at finetuning time, its weights are copied into a "target" network, except for the classification layer, which is randomly initialized, and finetuned on the target classes.[9] Additionally, the authors propose to regularize the finetuning stage by penalizing the activation of background features with L_2 loss (Background Depression), and another "Transfer-Knowledge" loss which pulls target network predictions closer to the source network predictions. Subsequently, TFA or Frustratingly Simple Few-Shot Object Detection [110] showed that even a simple finetuning approach with minimal modifications to Faster R-CNN can actually yield competitive performance for FSOD. TFA replaces the fully-connected classification heads of Faster R-CNN with *cosine* similarities; the authors argue that such feature normalization leads to reduced intra-class variance, and less accuracy decrease on the base classes. First, a Faster R-CNN with cosine classification heads is trained on the base classes using the usual loss. During few-shot finetuning, new classification weights are randomly initialized for novel classes, appended to the base weights, and the last layers of the model are finetuned on the base+novel classes, while keeping the backbone and

7. We only included methods evaluated on at least one of the dominant FSOD benchmarks/splits.
8. Use colors for quick reference to Table 3.
9. Slightly different from the FSOD framework presented in Section 4.1, the authors exclusively consider a cross-dataset scenario; therefore, target classes may or may not include sources classes.

TABLE 3
Few-Shot Object Detection methods with results on PASCAL VOC and MS COCO. Methods are categorized as finetuning -only, prototype -based, and modulation -based. TIP is a general add-on strategy for two-stage detectors. Faster RCNN+FT numbers are from TFA [110]. RepMet and Attention-FSOD numbers are from Meta-DETR [126]. Methods are sorted by MS COCO 30-shot nAP. Find the most up-to-date table at https://github.com/gabrielhuang/awesome-few-shot-object-detection

		VOC TFA-split (nAP50)			VOC Kang-split (nAP50)			MS COCO (nAP)	
Name	Type	1-shot	3-shot	10-shot	1-shot	3-shot	10-shot	10-shot	30-shot
LSTD [14]	finetuning	-	-	-	8.2	12.4	38.5	-	-
RepMet [55]	prototype	-	-	-	26.1	34.4	41.3	-	-
Meta-YOLO [54]	modulation	14.2	29.8	-	14.8	26.7	47.2	5.6	9.1
MetaDet [113]	modulation	-	-	-	18.9	30.2	49.6	7.1	11.3
Meta-RCNN [121]	modulation	-	-	-	19.9	35.0	51.5	8.7	12.4
Faster RCNN+FT [110]	finetuning	9.9	21.6	35.6	15.2	29.0	45.5	9.2	12.5
ACM-MetaRCNN [116]	modulation	-	-	-	31.9	35.9	53.1	9.4	12.8
TFA w/fc [110]	finetuning	22.9	40.4	52.0	36.8	43.6	57.0	10.0	13.4
TFA w/cos [110]	finetuning	25.3	42.1	52.8	39.8	44.7	56.0	10.0	13.7
Retentive RCNN [31]	finetuning	-	-	-	42.0	46.0	56.0	10.5	13.8
MPSR [115]	finetuning	-	-	-	41.7	51.4	61.8	9.8	14.1
Attention-FSOD [29]	modulation	-	-	-	-	-	-	12.0	-
FsDetView [119]	modulation	24.2	42.2	57.4	-	-	-	12.5	14.7
CME [67]	finetuning	-	-	-	41.5	50.4	60.9	15.1	16.9
TIP [66]	add-on	27.7	43.3	59.6	-	-	-	16.3	18.3
DAnA [17]	modulation	-	-	-	-	-	-	18.6	21.6
DeFRCN [87]	prototype	-	-	-	53.6	61.5	60.8	18.5	22.6
Meta-DETR [126]	modulation	20.4	46.3	57.8	-	-	-	17.8	22.9
DETReg [6]	finetuning	-	-	-	-	-	-	18.0	30.0

RPN frozen. MPSR [115] is also a finetuning approach. They propose an even more scale-aware Faster RCNN by combining the Feature Pyramid Network with traditional object pyramids [1]. After training the model on the base classes, the classification layer is simply discarded, a new classification layer is initialized, and the model does few-shot finetuning on the base+novel classes without freezing any layers. RetentiveRCNN [31] extend TFA to generalized FSOD, where the goal is to perform well on the novel classes without losing performance on the base classes. They observe a "massive variation of norms between base classes and unseen novel classes", which could explain why using cosine classification layers is better than fully connected ones. After training a Faster RCNN on the base classes, they freeze the base RPN and detection branches, and in parallel they introduce new finetuned RPN and detection branches to detect both base and novel classes. They also use a consistency loss, similar in spirit to LSTD's transfer-knowledge loss, to make predictions on base objects more similar in the base/base-and-novel branches. DETReg [6] use a finetuning approach on the Deformable DETR [132] architecture, and achieve state-of-the-art results on few-shot COCO object detection after proposing a self-supervised strategy for pretraining the detection heads, which is discussed in more depth in Section 5.3.1.

4.5.2 Conditioning-based methods

For clarity, we will refer to the image to process (to detect objects from) as the **query image**. In addition to the query image, conditioning-based methods are also fed with annotated **support images**, which are reference examples of each class to detect. Each support image generally has a single bounding-box around the object to detect. In the context of the FSOD framework presented in Section 4.1, support images are randomly sampled from all base classes during training (step 3), while a predefined (few-shot) set of base and novel images are used during finetuning and evaluation (steps 4 and 5). In this section, we review two types of conditioning-based methods: prototype-based, and modulation-based.

4.5.2.1 Prototype-based methods: RepMet [55], which stands for representative-based metric learning, is based on Faster-RCNN with Deformable Feature Pyramid Network (FPN), and learns representatives/prototypes for each object category. At base-training time, RepMet samples several supporting examples for each class, computes their ROI-pooled representations (representatives), and classifies object proposals according to their distance to the representatives. The gradients are propagated through both the proposals and the prototypes. At few-shot finetuning and evaluation time, representatives for the novel classes are computed, and objects proposals are classified using those new representatives. Optionally, the authors propose to finetune the novel prototypes by maximizing the detection loss on novel objects, which they find beneficial (denoted as "episode fine-tuning" in Table 3 of Karlinsky et al. [55]). ACM-MetaRCNN [116] have also proposed a baseline which combines Faster R-CNN with prototypical networks by replacing the classification layer with the non-parametric prototypical network equivalent. They investigate this base-

line with and without finetuning, and find that it is always beneficial to finetune.

4.5.2.2 Modulation-based methods: Modulation-based methods generally compute **support weights** (also known as *class embeddings*, *weights*, *prototypes*, or *attentive-vectors*) from the support features using a separate **conditioning branch** (sometimes called *reweighting* module [54], *guidance* module [66], or *remodeling* network [121]). Each class has its own support weights, which are usually computed by applying global average-pooling to the support features and have a shape $1 \times 1 \times C$ shape, where C is the number of channels. The support weights then are multiplied channel-wise with the query features to obtain class-specific query features, in a process known as **modulation** or **aggregation**. Finally, binary detections are predicted (*object* vs. *background*) on each set of class-specific features, and the results are merged to get multiclass detections. For faster inference, support weights can be precomputed and stored during the finetuning/conditioning step.

To the best of our knowledge, one of the first modulation-based method was Meta-YOLO [54], which is also the work that introduces the standardized FSOD splits (see **Figure 7**). Meta-YOLO uses both conditioning and finetuning. The model uses a feature extractor module to obtain the image features of the query image, and a reweighting module that extracts features from the bounding boxes of the support images to obtain reweighting vectors. The object mask binary matrices are added as an extra layer to the RGB images to form 4-channel images which are fed to the reweighting module. These vectors are used to condition the image features which allows the prediction layer to output the bounding boxes corresponding to the classes represented by those vectors. During base training, the two modules and the prediction layer are trained on the base classes. During few-shot finetuning, the model is finetuned using the K support examples per class, from the base and novel classes.[10] At few-shot evaluation time, reweighting vectors for each class are precomputed by averaging the corresponding vectors, and used to modulate the main features.

Since then, several improved conditioning-based methods have been introduced. Due to using different architectures and design choices, these works employ a diverse range of modulation strategies, which we discuss below. In **Meta-YOLO** [54], which is based on the single-stage detector YOLO, the features f_{qry} output by the backbone are directly multiplied channel-wise by the class embeddings f_{cls}, resulting in the modulated features $[f_{qry} \bigotimes f_{cls}]$. In **Meta-RCNN** [121] which is based on the two-stage detector Faster RCNN, the features are only multiplied after they have been pooled with RoIAlign; the consequence is that the Region Proposal Network (RPN) is agnostic to the object category. In **ACM-MetaRCNN** [116], also based on Faster RCNN, the feature maps are multiplied twice by the class prototypes: before running the RPN, and after pooling the features of each box, which means the RPN may produce different region proposals for each class. In **Attention-FSOD** [29], based on Faster RCNN, the query features are convolved channel-wise with support features (used as kernels) before feeding them to the RPN. In practice the authors find 1×1 support features to be optimal, and the convolution reduces to a channel-wise multiplication. The proposals are fed to a relational detection head, which classifies objects using matching scores by comparing query features with support features. Additionally, the authors explore three ways to model support-to-query relations in the detection head: globally, locally, and patch-wise. They find it beneficial to use all three types of relation heads. In **Fully Guided Network (FGN)** [30], based on Faster RCNN, the support features are global-average-pooled and averaged for each class to obtain the class-attentive vectors. Then, the query features are multiplied channel-wise with the class-attentive vectors and fed to the RPN to obtain class-specific proposals, which are aggregated between the different classes. Finally, in the relational detection head, the aligned query features and N support class averages are concatenated altogether and fed into a MLP to obtain box and multiclass predictions. Relational detection heads [101] have the ability to jointly predict boxes for all classes, which differs from other conditioning-based approaches which predict boxes independently for each class by relying on class-specific modulated features. In **FsDetView** [119], the query features are modulated by the support weights after ROI pooling. Instead of simply multiplying the features together, the authors propose to also concatenate and subtract them, resulting in the modulated features $[f_{qry} \bigotimes f_{cls}, f_{qry} - f_{cls}, f_{qry}]$. **Meta-DETR** [126] adapts FsDetView's modulation strategy to the DETR architecture [9]. At the output of the backbone, both support and query features are fed to the transformer encoder, then the query features are multiplied with global-average-pooled support vectors, and fed to the transformer decoder to make binary predictions.

4.5.2.3 Necessity of finetuning: Only a few FSOD methods such as Li et al. [70] or Attention-FSOD [29] present themselves as methods that do not require finetuning. However, we should note that most if not all of the conditioning-based methods presented in the previous section could technically be used without finetuning on base+novel classes, by directly conditioning on the support examples at few-shot evaluation time. In practice, most works find it beneficial to finetune, and in fact many of the conditioning-based methods reviewed above do not even report numbers without finetuning. For instance, ACM-MetaRCNN, a conditioning-based model, finetune their model, except for 1-shot and 2-shot on Pascal VOC, where they do not finetune "to avoid overfitting". Even Attention-FSOD [29], which claims to be usable without finetuning, achieves its best performance after finetuning (see for instance Table 4).

4.5.3 Add-on methods

Some FSOD methods do not propose a specific architecture, but instead propose add-on tricks that can be combined with many of the existing FSOD methods to boost performance. For instance, Transformation Invariant Principle (TIP) [66] is a regularization strategy based on data augmentation transformations, which can be applied to any two-stage FSOD method. Specifically, TIP proposes to minimize several consistency losses: the guidance vectors (class weights)

10. Since only K labeled bounding boxes are available for the novel classes, to balance between samples from the base and novel classes, only K boxes are included for each base class

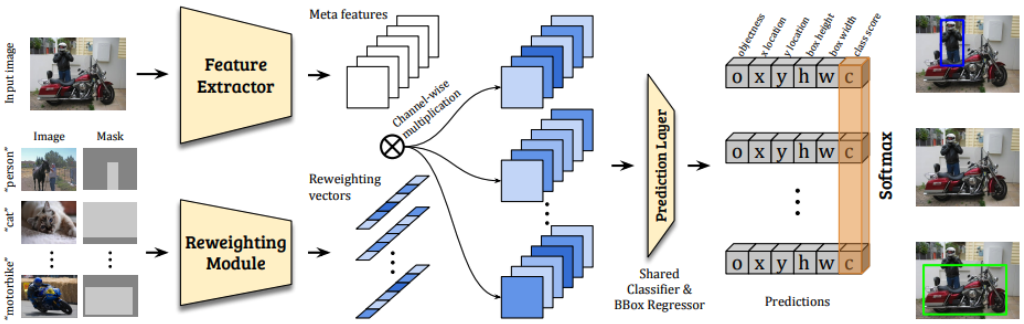

Fig. 7. **Meta-YOLO**, a modulation-based FSOD method.

for a support object (first view) and its transformed version (second view) should be close in feature space (the authors find the L2 distance to give best results). Additionally, TIP pools features from one view using proposals generated from another view; the resulting detections are used to compute another detection loss (regression and classification).

5 Self-Supervised Pretraining

Until recently, the standard approach in deep object detection was to pretrain the backbone on supervised ImageNet [24] classification. This still holds for modern iterations of two-stage detectors such as Faster R-CNN [95] – as per its `detectron2` implementation – as well as one-stage detectors such as YOLOv3 [93], SSD [75], RetinaNet [73] and recent transformer-based detectors like DETR [9] and Deformable-DETR [132].

Self-supervised pretraining has emerged as an effective alternative to supervised pretraining where the supervision comes from the data itself. The key idea is to automatically generate labels from unlabeled data, and learning to predict those labels back. This process is known as solving a pretext task. For instance, a common pretext task is to predict the relative position of two random crops from the same image [85]. This broad definition could potentially include many unsupervised methods such as VAEs [57] and GANs [38, 48] but in practice the self-supervised term is used for methods for which the pretext task differs from the downstream task [15, 39, 44]. Some language models such as word2vec [79] are also considered to be self-supervised.

Starting with SimCLR [15] and MoCo [44], people have experimented initializing object detection backbones with unsupervised representations learned on ImageNet (or COCO) instead of supervised ones. Since the pretext tasks are fairly general, there is the hope that unsupervised representations might generalize better to downstream tasks than classification-based ones. Recent works [6, 111, 114, 122] which we will refer to as *self-supervised object detection* methods go beyond backbone-pretraining by also pretraining the detection heads specifically for object detection.

In Section 5.1 we review self-supervised *classification* methods; then in Section 5.2 we discuss their limitations for initializing object detection backbones; finally in Section 5.3 we review self-supervised *object detection* approaches, which unlike the previous methods, are specifically tailored to object detection.

5.1 Backbone-only Pretraining

In the image classification domain, self-supervised learning has emerged as a strong alternative to supervised pretraining, especially in domains where images are abundant but annotations are scarce [78]. Self-supervised classification methods are not limited to classification, as the learned feature extractor can used to initialize the backbone of common object detection architectures. We categorize backbone pretraining strategies into constrastive, clustering-based and self-distillative. We will omit reconstruction [85, 107, 127, 128] methods and visual common sense [26, 35, 82] based tasks, as to our knowledge, they have not been used for object detection.

5.1.1 Contrastive Learning ✗ [11]

These approaches leverage the InfoNCE [83] loss to compare pairs of samples which can be positive pairs or negative pairs. Usually, positive pairs are generated from different *views* (data augmentations) of the same image, while negative pairs are generated from different images. Contrastive Predictive Coding (CPC) is one of the first approaches to be competitive with respect to supervised classification [45, 83]. In CPC, the goal is to predict a future part of a sequential signal $p(x|c)$ given some previous context of that signal (c). Since reconstructing x from c is difficult in high-dimensional spaces, they propose a contrastive objective instead. Given a set of random samples, containing one *positive* sample $x^+ \sim p(x|c)$, and $N-1$ *negative* samples $x_1^-, \dots, x_N^- \sim p(x)$ from the "proposal" distribution, they propose to learn a function $f_\theta(x, c)$ which minimizes the InfoNCE loss:

$$\mathcal{L}_{\text{InfoNCE}}(\theta) = -\mathbb{E}_X \left[\log \frac{f_\theta(x^+, c)}{\sum_i f_\theta(x_i^-, c)} \right]. \quad (1)$$

The density ratio $f_\theta(x, c)$ can be interpreted as an affinity score, which should be high for the real sample and low for negative samples.

More recent approaches such as momentum contrast (MoCo) [20, 21, 44], or SimCLR [15, 16] reinterpret the InfoNCE loss in a different context. Given a reference image x_0, positive samples $x^+ \sim p(x|x_0)$ are generated using data augmentation on x_0, and negative samples $x_1^-, \dots, x_N^- \sim$

11. Use colors for quick reference to Table 4. The ✗ means backbone-only pretraining.

$p(x)$ are other images sampled from the dataset. The InfoNCE loss becomes:

$$\mathcal{L}_{\text{InfoNCE}}(\theta) = -\mathbb{E}_X \left[\log \frac{f_\theta(x^+, x_0)}{\sum_i f_\theta(x_i^-, x_0)} \right]. \quad (2)$$

The goal is to learn representations which maximize the affinity $f_\theta(x, x_0)$ between different data-augmentations (**views**) of the same image, and minimize the affinity between different images.

While both MoCo and SimCLR find crucial to use a large set of negative examples, they differ in their approach to obtain them. SimCLR uses a large batch size, while MoCo uses smaller batches but stores the embeddings in a queue, which is used to retrieve negative examples. To prevent drift between queued embeddings and positive examples, MoCo encodes negative examples with an exponential moving average of the weights.

Most self-supervised pre-training methods in the literature aim to learn a global image representation to transfer to a given downstream task [12, 13, 15, 20, 21, 44]. However, global image representations might not be optimal for dense prediction tasks such as detection and segmentation. In order to bridge the gap between self-supervised pre-training and dense prediction, Pinheiro et al. [86] proposed VADeR, a pixel-level contrastive learning task for dense visual representation. Different from global contrastive learning approaches such as SimCLR, VADeR uses an encoder-decoder architecture. Then, given the output feature maps for a positive sample, an augmented sample, and a negative sample, the InfoNCE loss is applied between the decoder's pixel features rather than being applied to the average of the decoder's output. As a result, VADeR achieves encouraging results when compared to strong baselines in many structured prediction tasks, ranging from recognition to geometry.

5.1.2 *Clustering-Based ✗* [12]

Clustering-based methods rely on unsupervised clustering algorithms to generate pseudo-labels for training deep learning models [120, 123]. A key idea is to alternate between clustering learned representations, and using the predicted cluster assignments to improve representations in return. Caron et al. [10, 11] show that k-means cluster assignments are an effective supervisory signal for learning visual representations. Asano et al. [4] show that cluster assignments can be solved as an optimal transport problem. Based on previous approaches, Swapping Assignments between multiple Views of the same image (SwAV) [12] was proposed. SwAV attempts to predict the cluster assignments for one view from another view (data augmentation) of the same image. It uses the Sinkhorn-Knopp algorithm for clustering [22], which has previously been explored for recovering labels in few-shot classification [49], and has good properties such as quick convergence and differentiability [22]. SwAV avoids trivial solutions where all features collapse to the same representation collapse by using the appropriate amount of entropy regularization.

12. Use colors for quick reference to Table 4. Checkmarks ✓ and ✗ indicate whether the detection heads are pretrained.

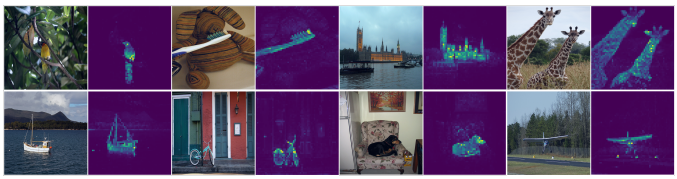

Fig. 8. **DINO's attention maps.** Since DINO is based on a visual transformer, the attention maps corresponding to the `[CLS]` token can be plotted. Despite being trained with no supervision, different attention heads are found to segment different objects. Source: Caron et al. [11].

5.1.3 *Knowledge Self-distillation (BYOL) ✗*

Self-distillative approaches such as Bring Your Own Latent (BYOL) [39] and mean teacher [104] move away from contrastive learning by maximizing the similarity between the predictions of a teacher and a student model. The student model is optimized using gradient descent, while the teacher model is instantiated as an exponential moving average of the student weights [39]. In order to prevent them from collapsing to the same representation, Grill et al. [39] found it helpful to use other tricks such as softmax sharpening and recentering. Subsequent approaches such as DINO [13] and EsViT [68] leverage vision transformers (ViT) [27] instead of residual networks, following the trend of using self-supervision in natural language processing [8, 88, 106]. They divide the input image into a grid of small patches (8×8 pixels for DINO) and feed them to a ViT. The last feature map, which could be used as a dense representation, is average-pooled into a single vector and compared between teacher and student models using cross-entropy. The main difference between the two is that EsViT uses a two-stage architecture and a part-based loss. The first improvement reduces the amount of image patches in the second stage, which makes the model more efficient. The second improvement introduces an additional loss besides DINO's teacher-student loss that encourages matching regions across multiple views to match their respective student and teacher representations.

Interestingly, the authors of DINO show that semantic segmentation masks naturally emerge from the attention masks of the visual transformer (see Figure 8). This is very surprising giving that the model was trained with no supervision whatsoever. This suggests that combining dense pretext tasks such as VADeR [86] and DINO [13] with attention-based models such as transformers could improve the transferability of self-supervised learning methods to dense downstream tasks.

5.2 Issues of Combining Self-supervised Classification with Detection

Conceptually, there are some issues with transferring classification-based representations to object detectors, whether the representations were supervised or self-supervised.

5.2.1 *Untrained Detection Heads*

The first issue is with untrained detection heads, due to the architecture mismatch. For instance, it is a common practice to combine Resnet-50 and Resnet-101 backbones with a

Feature Pyramid Network (FPN). However, the FPN is initialized from scratch and does not benefit from pretraining. The same goes for the Region Proposal Network (RPN) and the detection regression and classification heads, in the case of Faster RCNN; and for the encoder and decoder, in the case of DETR-style architectures.

5.2.2 Task Mismatch

Secondly, ImageNet top-1 classification accuracy does not necessarily correlate with object detection performance. While certain properties such as translation and scale invariance are desirable for classification, they might actually hinder object localization [81, 117]. Classification-based representations such as MoCo [44] or SimCLR [15] might discard spatial information that is useful for localization, because they are irrelevant for solving the pretraining tasks. Moreover, data augmentation strategies such as random cropping and jittering may introduce undesirable invariances into the network. In fact, Yang et al. [122] show that MoCo can perform better on object detection than BYOL and SwAV, despite having worse classification performance.

5.3 Pretraining the Detection Heads

Self-supervised object detection approaches attempt to remedy those issues by pretraining the object detection pipeline on a variety of unsupervised pretext tasks.

5.3.1 Predictive approaches ✓

Predictive approaches such as UP-DETR [23] and DETReg [6] pretrain the detection heads of DETR by making them re-predict the position of automatically generated "ground-truth" crops. These crops are generated either randomly for UP-DETR or using Selective Search [105] for DETReg, a training-free heuristic based on iteratively merging regions with similar colors, textures, and other local characteristics. [13]

To pretrain DETR on unsupervised images, the multi-class classification heads (see Figure 3) which would normally predict the object category or *background* are replaced with a *binary* classification head. In DETReg, the goal is to detect the top proposals generated by Selective Search, as if they were ground-truth foreground objects. The usual DETR loss is used, except that the matching cost used to compute correspondences between ground-truth boxes and detections is a function of the predicted binary labels and locations – instead of ground truth and predicted multiclass labels. In UP-DETR, the goal is to predict back the positions of the random crops. Specifically, the transformer decoder is conditioned on the random crops by adding their corresponding features to the decoder input (see *object queries* in Figure 3). This is done by partitioning the object queries into K groups,[14] and adding a different random crop to each group. The loss is computed by finding the optimal matching between the predicted boxes and the "ground-truth" random crops using the Hungarian algorithm [80], where the cost of matching two boxes is a function of their location and predicted binary label.

On top of the DETR loss, an additional reconstruction loss is used to force the decoder transformer to reconstruct its input. DETReg uses a simple L1 loss $\mathcal{L}_{rec}(z_i, z_j) = ||z_i - z_j||_1$, while UP-DETR uses cosine similarity $\mathcal{L}_{rec}(z_i, z_j) = \left\| z_i/\|z_i\| - z_j/\|z_j\| \right\|_1$. Also, an interesting by-product of UP-DETR is that it learns a conditioning branch, which can be reused directly for one-shot object detection by replacing the random crops with support images. The paper provides some results on Pascal VOC [23].[15]

5.3.2 Contrastive approaches ✓

Contrastive approaches such as DenseCL [111], InsLoc [122] and ReSim [118] attempt to train the box heads of Faster RCNN to output a representation such that *positive* pairs are close and *negative* pairs are far apart, by minimizing the InfoNCE loss. Before describing what constitutes positive/negative pairs, we can further categorize these methods into crop-level and dense approaches.

In **crop-level constrastive** methods such as InsLoc [122], features for each crop are computed using RoIAlign [95], then transformed by the RoI heads to a single $1 \times 1 \times d$ vector. Positive pairs are generated by randomly cropping two views of the same image, while negative pairs are generated by using different images. Specifically in InsLoc, positive pairs are generated by pasting two views of the same object (the foreground) at random locations and scales onto other images of the dataset (the background). The authors also introduce a cut-and-paste scheme in order to force the receptive field of RoIAlign to ignore distractor features outside the bounding box.

In **dense constrastive** approaches such as DenseCL and ReSim, pairs of feature maps are compared based on the spatial correspondance between them. In ReSim [118], two overlapping crops are generated from two different views of the same image. Then, a sliding window is moved across the overlapping area, and the pooled representations are compared at each of the final convolutional layers. Positive pairs consist of aligned sliding windows across two views of the same image, while negative pairs either consist of unaligned sliding windows, or sliding windows from two different images. In DenseCL [111], instead of using spatial correspondence, positive pairs are generated by matching each feature from one view to the feature with highest cosine similarity in another view of the same image. Negative examples are simply features from different images. Additionally, the authors combine this dense loss with a global MoCo-style loss, which they claim is necessary to bootstrap correct correspondences.

5.3.3 Self-distillative approaches (BYOL) ✓

Self-distillative (BYOL-based) approaches such as SoCo [114] depart significantly from contrastive approaches as there is no need for negative examples. Selective object COnstrastive learning (SoCo) builds on top of BYOL [39]

13. Note that Selective Search used to be a popular training-free heuristic for generating high-recall low-precision region proposals, and was used in RCNN [37] and Fast RCNN [36] before it was replaced by a trained Region Proposal Network (RPN).

14. The way UP-DETR matches predictions and ground-truth from different groups instead of matching only within the same groups might not necessarily be an intended feature, but rather a consequence of building on top of existing DETR code. In practice, this does not make much difference as the groups are matched correctly (personal communication with the authors).

15. Not reported in Table 4 due to using different splits.

TABLE 4
Comparison of Self-Supervised Object Detection Methods pretrained on unlabeled ImageNet. The check marks ✓ and ✗ refer to whether the methods pretrain both the backbone and detection heads vs. only the backbone.

Name	Pretrains Detector	Loss	View Matching	Region Proposal Mechanism	Pascal AP50	COCO AP	Best Backbone	Object Detector
DETReg [6]	✓	predictive ✓	-	selective search	83.3	45.5	R50	Def.DETR
SoCo [114]	✓	BYOL ✓	crop	selective search	83.8	44.3	R50-FPN	F-RCNN
InsLoc [122]	✓	contrastive ✓	crop	random crop	83.0	43.3	R50-C4	F-RCNN
UP-DETR [23]	✓	predictive ✓	-	random crop	80.1	42.8	R50-C4	DETR
ReSim [118]	✓	contrastive ✓	sliding window	random crop	83.1	41.4	R50-FPN/C4	F-RCNN
DenseCL [111]	✓	contrastive ✓	feature	random crop	82.8	41.2	R50-FPN/C4	F-RCNN
VaDer [86]	FPN-only	contrastive ✗	feature	-	-	39.2	R50-FPN	F-RCNN
EsViT [68]	✗	BYOL ✗	image	-	-	46.2	Swin-ViT	F-RCNN
DETcon [46]	✗	contrastive ✗	mask	grid/FH/MCG	82.8	43.4	R50-FPN	F-RCNN
BYOL [39]	✗	BYOL ✗	image		81.0	42.3	R50-FPN/C4	F-RCNN
DI✗ [13]	✗	BYOL ✗	image	-	-	-	ViT/R-50	-
SwAV [12]	✗	clustering ✗	image	-	77.4	42.3	ResNet	F-RCNN
MoCo [20]	✗	contrastive ✗	image	-	82.5	41.7	R50-FPN/C4	F-RCNN
SimCLR [15]	✗	contrastive ✗	image	-	81.9	39.6	ResNet	F-RCNN

and pretrains both the backbone, feature pyramid, and RoI heads of a FPN-based Faster RCNN by training two networks simultaneously. The "student" network f_θ uses gradient descent to copy the feature maps of a "teacher" network f_ξ, which is an exponential moving average of the student network, and the student network is optimized using gradient descent. For a given image, object proposals (denote the bounding boxes b) are generated unsupervisedly using Selective Search [105]. Then, two views V_1, V_2 are generated and respectively fed into student and teacher FPNs to get feature maps v_1, v_2. Box features are computed for each bounding box b by pooling v_1, v_2 with RoIAlign and passing them to the RoI heads:

$$h_1 = f_\theta^H(\text{RoIAlign}(v_1, b), \quad h_2 = f_\xi^H(\text{RoIAlign}(v_2, b).$$

The box features h_1, h_2 are then projected to obtain latent embeddings e_1, e_2, and their cosine similariy is minimized

$$\mathcal{L}(\theta) = -\frac{\langle e_1, e_2 \rangle}{||e_1||_2 \cdot ||e_2||_2}.$$

In practice, SoCo introduces several other tricks, such as using more than two views and resizing them at multiple scales, jittering the proposed box coordinates, and filtering proposals by aspect ratio [114].

5.4 Comparison of Self-Supervised Object Detection Methods

In Table 4, we review the self-supervised object detection methods discussed previously , and report their performance on Pascal VOC and MS COCO object detection (non few-shot). Note that the numbers are not directly comparable in absolute value, due to variations in model architectures, hyperparameters, learning rate schedules, data augmentation schemes, and other implementation details. Instead, we encourage the reader to dig into the corresponding ablation studies of those works.

This table contains methods specifically geared towards object detection as presented in Section 5.3, which train the backbone ("Pretrains Detector: Yes"), and general purpose representations as presented in Section 5.1 which only pretrain the backbone on top of which an object detector was fitted a posteriori, often by a subsequent work ("Pretrains Detector: No"). For instance, most of the numbers for DenseCL, BYOL, DETcon, MoCo, SimCLR and SwAV are taken from the SoCo paper [114]. VaDer is in between, as it does not pretrain detection heads but does pretrain a feature pyramid network (FPN) alongside the backbone.

The methods can be categorized into four types of losses: self-distillative (BYOL), predictive (predictive), constrastive (contrastive) and clustering-based (clustering). The check marks ✓and ✗refer to whether these methods also pretrain the detection heads. **View Matching** refers to the way different views of the same image are matched. From most global to most local: image > crop, mask > sliding window > feature. *Region proposal mechanism* describes how unsupervised regions are generated for the purpose of pretraining (not to be confused with the candidate proposals in two-stage detectors). DETcon generates masks from: "grid" a fixed-size grid, "FH" the Felzenszwalb-Huttenlocher algorithm [32], or "MCG" Multiscale Combinatorial Grouping [3]. "R50-FPN" means ResNet-50 with Feature Pyramid Network. "R50-C4" means using ResNet-50's C4 layer. "ViT" is the visual transformer [27]. "Swin" is a type of hierarchical visual transformer [76]. "F-RCNN" stands for Faster R-CNN, "Def. DETR" for deformable DETR.

Many FSOD methods [39, 44, 111, 118] are found to perform better using multi-scale features with a FPN for MS COCO, but with single-scale C4 features for Pascal VOC, which may be a consequence of the limited size of Pascal.

6 Takeaways & Trends

We discuss our main takeaways and forecasted trends from this survey.

6.1 Finetuning is a strong baseline

Almost every few-shot object detection method we have reviewed finetunes on the novel classes. This is the case even for conditioning-based methods, which could technically be used without finetuning by conditioning on the support examples, but have been found to benefit from finetuning anyways. The problem is that finetuning approaches are slower and may require more hyperparameter tuning. This could be a serious obstacle to deploying such methods in the real world. In general, we hope to more see competitive finetuning-free methods in the future.

6.2 Impact of self-supervision for object detection

It is somewhat surprising that self-supervised object detection pretraining only brings limited improvements for traditional object detection. This could be explained by the fact that post-pretraining, the object detector is finetuned on the labeled dataset, which could render self-supervision redundant. It could also be that current experiments are mainly limited to ImageNet and MS COCO pretraining, whilst self-supervision could potentially benefit from larger unlabeled datasets. However, the impact of self-supervised pretraining seems to be more significant for few-shot and low-data object detection. In fact, the state-of-the-art FSOD results on MS COCO are from DETReg [6], a self-supervised object detection method.

6.3 Using heuristics to generate weak labels

A general trend in self-supervised learning is to use heuristics to generate weak or noisy labels. Data augmentations are now widely used for generating positive pairs in the context of self-supervised classification [15, 39, 44]. Specifically to object detection, DETReg [6] and SoCo [114] have adopted Selective Search [105], for generating crops which are more likely to contain objects. On the other hand, DetCon [46] have explored using the Felzenszwalb-Huttenlocher algorithm and Multiscale Combinatorial Grouping to generate better segmentation masks for feature pooling. Since these heuristics come from traditional computer vision, we expect practitioners to continue adapting more of them to improve self-supervised training in the future. An important question is how such heuristics can be integrated in an iterative bootstrapping procedure: as better representations are learned, it might be worthwhile to gradually replace the initial heuristics with learned and improved ones (e.g. replacing selective search with a learned RPN). One possible direction of research could be to develop differentiable/learnable versions of these traditional heuristics.

6.4 Rise of transformers

Visual transformers have gained increasing traction in object detection, both as backbones and as end-to-end detection heads. For using them as backbones, works such as DINO [13] have shown that fully unsupervised pretraining of visual transformers can lead to the emergence of object segmentation capabilities. Specifically in their case, the multi-head attention modules learn to segment foreground objects as a byproduct of solving the pretext task, as shown in Figure 8. More generally, there is growing belief from the study of scaling laws for foundational models that visual transformers can generalize better than ResNets to large scale training [125]. Some self-supervised methods, such as EsViT [68], also rely on recent iterations of visual transformers such as Swin [76] to obtain state-of-the-art results. When using them as detection heads, DETR [9] has shown that transformer-based detection heads can be trained end-to-end. In particular, they are capable of making joint predictions and dealing with redundant detections, thus removing the need to rely on heuristics such as non-maximum suppression (NMS). More recent work such as Pix2Seq [18] has shown that object detection can be formulated as a language modeling task. The flexibility that comes with language modeling could blur the line between pure vision tasks (such as object detection) and vision-to-language tasks (such as image captioning or visual question answering), lead to simpler architectures, more end-to-end approaches with less heuristics (such as NMS), and pave the way to foundational models for vision and language tasks.

6.5 Problems with current evaluation procedures

Comparisons such as Table 3 and Table 4 should only be used to get a general idea of the performance of those systems. The numbers themselves often not directly comparable, due to variations in backbone architecture, use of multi-scale features (FPN), varying detection architectures, types of data augmentations used, learning rate scheduling, or even things as trivial as input image size resizing.

6.5.1 Differences in implementation details

In fact, many of the modulation-based FSOD methods we have reviewed in Section 4.5.2.2 have quite a similar structure, differing only in the modulation strategy, backbone architecture and object detector used. This raises the question of how much of the performance of state-of-the-art methods is owed to new ideas rather than better hyperparameter tuning or using better architectures. One way would be to see how much performance we can get with running older methods in newer frameworks, or building an unifying benchmark.

6.5.2 Issues with data splits

Specifically to the FSOD use of Pascal VOC, there has been a shift from using Kang's splits to TFA's splits which were introduced later to alleviate the variance problems with Kang's splits. Despite the fact that the two splits can yield wildly different numbers (see for instance the line on TFA w/cos), several works mistakenly mix them up in the same tables [66, 119, 126]. More generally, the fact that virtually every FSOD paper – regular object detection papers too– trains on the union of training and validation sets (trainval) and uses the test set for hyperparameter tuning can lead to overfitting and overestimating the actual generalization performance.

6.5.3 Proposed guidelines

Therefore, we propose the following guidelines for having more comparable results:

1) Define and use proper train/val/test splits. Researchers should agree on newer splits or benchmarks, as no single researcher has any incentive to stop overfitting on the test set.
2) Do not propose benchmarks which are prone to high variance, such as Kang's splits for Pascal or TFA splits for LVIS. Prior work on few-shot classification has consistently provided confidence intervals by averaging results over multiple episodes, and sampling the few-shot training set instead of fixing the instances [19, 99, 108].
3) Standardize implementation details such as image resizing, whitening, and data augmentations. Define standard backbone and detector architectures to be explored for each benchmark. Results could be presented in two categories: fixed architecture and best architecture. The introduction of Detectron2 has already led to more standardization and code sharing, providing among other things a standard implementation of Faster R-CNN with FPN.[16]
4) Relate new tasks to existing tasks. For instance, the dominant FSOD and few-shot classification frameworks use different terminologies, training and evaluation procedures, and FSOD could have benefited from FSC best practices. We found it necessary to clarify the differences and subtleties in Section 4.2.

7 Related Tasks

We briefly discuss other related tasks, which are out of the scope of this survey.

7.1 Weakly-supervised object detection

Image-level and point-level annotations are cheaper and faster to obtain, and noisy image-level labels could even be generated automatically using image search engines. Weakly supervised object detectors are trained using only image-level annotations without requiring bounding boxes [7, 52, 103] and could therefore benefit from a larger pool of labels. Many weakly supervised detection methods fall under multiple-instance learning (MIL) [25] where each image corresponds to a bag of object proposals. A bag is given the class label based on whether the label exists in the image. Li et al. [69] present a two-step approach that includes selecting good object proposals; then training a Faster RCNN [95]. Tang et al. [102] use a refinement learning strategy to select good quality proposals. C-MIL [109] introduces an optimization method to avoid selecting poor proposals, C-WSL [34] uses object count information to obtain the highest scoring proposals, WISE [62] that uses class activation maps to score proposals and LOOC [63] can be used to detect objects in crowded scenes. Point-level annotations can also be use for object detection like in [61, 64, 65], but they require slightly more human effort.

16. Despite the valuable efforts of Detectron2 towards standardization and open-sourcing, we did find the framework overwhelming for some use-cases. This resulted in additional difficulties when working with Detectron2-based projects due to the highly abstract nature of the framework. We hope that future frameworks will be more user-centric. For instance, a micro-framework with independently-usable modules might lead to more readable user code.

7.2 Self-supervision using other modalities

Instructional videos are a natural source of self-supervision, as they contain both speech and visual information. For instance, Amrani et al. [2] recently proposed using unlabeled narrated instructional videos to learn an object detector by exploiting the correlations between narration and video. They start by extracting video transcripts with an external method. For a given object, they generate positive frames from the temporal period where the object is mentioned, and negative frames from videos that do not mention the object. They extract bounding boxes using Selective Search [105], compute their features using a pretrained backbone, cluster them in feature space, assign a score to each cluster, and filter out the noisiest examples. Finally, they train a detector on the remaining bounding boxes. Laradji et al. [60] deal with 3D ct scans using self-supervision and weak supervision to train a model to predict regions in the lungs that are infected with COVID. Liu et al. [74] used self-supervision by making sure outputs are consistent between different viewpoints and augmentations which in turn helped improve 2D to 3D reconstruction.

7.3 Low-data object detection

Low-data object detection (LSOD) is closely related to few-shot object detection (FSOD). Instead of having a distinction between base classes (many examples) and novel classes (few-shot), LSOD assumes that the number of examples for all classes is limited, which is generally simulated by considering a fraction of the labels of traditional object detection datasets (e.g by taking 1%, 5%, 10% of MS COCO). Those methods typically pretrain on larger amount of unlabeled data. Several of the self-supervised object detection methods reviewed in this survey also report numbers in the low-data regime. See for instance the results on *mini*COCO of DETReg [6], SoCo [114], InsLoc [122], DenseCL [111].

7.4 Few-shot semantic segmentation

Few-shot object detection has many similarities to few-shot semantic segmentation as they both require us to identify the objects of interest in the images. However, semantic segmentation only considers the class of individual pixels and does not require to identify the individual objects in the image. Semantic segmentation models tend to be simpler as they are usually based on architectures that only have a downsampling path and an upsampling path [77, 97], as opposed to a proposal generator and additional networks for classification and regression [36]. Further, people have explored few-shot semantic segmentation using weaker labels than the full segmentation masks. For instance, Rakelly et al. [91], Siam et al. [98] allow annotators to label only few pixels per object in the support or image-level labels for meta-testing. Using weakly supervised methods for few-shot object detection is is an interesting direction that is fairly unexplored.

7.5 Zero-shot object detection

Zero-shot object detection and instance segmentation are about learning to detect (resp. segment) novel objects based on a non-visual descriptions of them. These descriptions

could be in the form of semantic attributes, such as "long tail" and "orange beak" in the case of bird classification. In practice, the attribute vector often consists of pretrained word embeddings, since those are readily available and contain implicit world knowledge from large unlabeled datasets. When using word embeddings, a common strategy is to modify existing object detection/instance segmentation heads by projecting object feature maps to have same dimensionality as word embeddings. This is the case of ViLD [40], ZSI [130], BLC [129], PL [90], DSES [5], who propose many tricks to improve performance, such as distilling pretrained vision-language models like CLIP [89], learning improved word embeddings for "foreground" and "background" for the RPN, using separate pathways for seen and unseen classes to avoid catastrophic forgetting, and explore different ways to mingle semantic and visual information. Overall, several innovations from zero-shot classification and object detection, few-shot object detection, and instance segmentation could be shared in the future.

8 Conclusion

We have formalized the few-shot object detection framework and reviewed the main benchmarks and evaluation metrics. We have categorized, reviewed, and compared several few-shot and self-supervised object detection methods. Finally, we have summarized our main takeaways, made future best practice recommendations, highlighted trends to follow, and given pointers to related tasks.

Acknowledgments

This research was partially supported by the Canada CIFAR AI Chair Program, the NSERC Discovery Grant RGPIN-2017-06936, by project PID2020-120611RB-I00/AEI/10.13039/501100011033 and by Mitacs through the Mitacs Accelerate program. Simon Lacoste-Julien is a CIFAR Associate Fellow in the Learning in Machines & Brains program.

References

[1] E. H. Adelson, C. H. Anderson, J. R. Bergen, P. J. Burt, and J. M. Ogden. Pyramid methods in image processing. *RCA engineer*, 29(6):33–41, 1984.

[2] E. Amrani, R. Ben-Ari, I. Shapira, T. Hakim, and A. Bronstein. Self-supervised object detection and retrieval using unlabeled videos. In *CVPR Workshops*, 2020.

[3] P. Arbeláez, J. Pont-Tuset, J. T. Barron, F. Marques, and J. Malik. Multiscale combinatorial grouping. In *CVPR*, 2014.

[4] Y. M. Asano, C. Rupprecht, and A. Vedaldi. Self-labelling via simultaneous clustering and representation learning. *arXiv:1911.05371*, 2019.

[5] A. Bansal, K. Sikka, G. Sharma, R. Chellappa, and A. Divakaran. Zero-shot object detection. In *ECCV*, 2018.

[6] A. Bar, X. Wang, V. Kantorov, C. J. Reed, R. Herzig, G. Chechik, A. Rohrbach, T. Darrell, and A. Globerson. Detreg: Unsupervised pretraining with region priors for object detection. *arXiv:2106.04550*, 2021.

[7] H. Bilen and A. Vedaldi. Weakly supervised deep detection networks. In *CVPR*, 2016.

[8] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners. In *NeurIPS*, 2020.

[9] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko. End-to-end object detection with transformers. In *ECCV*, 2020.

[10] M. Caron, P. Bojanowski, A. Joulin, and M. Douze. Deep clustering for unsupervised learning of visual features. In *ECCV*, 2018.

[11] M. Caron, P. Bojanowski, J. Mairal, and A. Joulin. Unsupervised pre-training of image features on non-curated data. In *CVPR*, 2019.

[12] M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In *NeurIPS*, 2020.

[13] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin. Emerging properties in self-supervised vision transformers. *arXiv:2104.14294*, 2021.

[14] H. Chen, Y. Wang, G. Wang, and Y. Qiao. Lstd: A low-shot transfer detector for object detection. In *AAAI*, 2018.

[15] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, 2020.

[16] T. Chen, S. Kornblith, K. Swersky, M. Norouzi, and G. E. Hinton. Big self-supervised models are strong semi-supervised learners. *NeurIPS*, 2020.

[17] T. Chen, Y. Liu, H. Su, Y. Chan, Y. Lin, J. Yeh, W. Chen, and W. H. Hsu. Dual-awareness attention for few-shot object detection. *IEEE TM*, 23, 2021.

[18] T. Chen, S. Saxena, L. Li, D. J. Fleet, and G. Hinton. Pix2seq: A language modeling framework for object detection. *arXiv:2109.10852*, 2021.

[19] W.-Y. Chen, Y.-C. Liu, Z. Kira, Y.-C. F. Wang, and J.-B. Huang. A closer look at few-shot classification. In *ICLR*, 2018.

[20] X. Chen, H. Fan, R. Girshick, and K. He. Improved baselines with momentum contrastive learning. *arXiv:2003.04297*, 2020.

[21] X. Chen, S. Xie, and K. He. An empirical study of training self-supervised vision transformers. *arXiv:2104.02057*, 2021.

[22] M. Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. *NeurIPS*, 2013.

[23] Z. Dai, B. Cai, Y. Lin, and J. Chen. Up-detr: Unsupervised pre-training for object detection with transformers. In *CVPR*, 2021.

[24] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[25] T. G. Dietterich, R. H. Lathrop, and T. Lozano-Pérez. Solving the multiple instance problem with axis-

parallel rectangles. *Artificial intelligence*, 89(1-2):31–71, 1997.

[26] C. Doersch, A. Gupta, and A. A. Efros. Unsupervised visual representation learning by context prediction. In *ICCV*, 2015.

[27] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*, 2021.

[28] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 88(2):303–338, 2010.

[29] Q. Fan, W. Zhuo, C.-K. Tang, and Y.-W. Tai. Few-shot object detection with attention-rpn and multi-relation detector. In *CVPR*, 2020.

[30] Z. Fan, J.-G. Yu, Z. Liang, J. Ou, C. Gao, G.-S. Xia, and Y. Li. Fgn: Fully guided network for few-shot instance segmentation. In *CVPR*, 2020.

[31] Z. Fan, Y. Ma, Z. Li, and J. Sun. Generalized few-shot object detection without forgetting. In *CVPR*, 2021.

[32] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *IJCV*, 2004.

[33] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017.

[34] M. Gao, A. Li, R. Yu, V. I. Morariu, and L. S. Davis. C-wsl: Count-guided weakly supervised localization. In *ECCV*, 2018.

[35] S. Gidaris, P. Singh, and N. Komodakis. Unsupervised representation learning by predicting image rotations. In *ICLR*, 2018.

[36] R. Girshick. Fast r-cnn. In *ICCV*, 2015.

[37] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

[38] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks. 2014.

[39] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. Richemond, E. Buchatskaya, C. Doersch, B. Avila Pires, Z. Guo, M. Gheshlaghi Azar, B. Piot, k. kavukcuoglu, R. Munos, and M. Valko. Bootstrap your own latent - a new approach to self-supervised learning. In *NeurIPS*, volume 33, pages 21271–21284, 2020.

[40] X. Gu, T.-Y. Lin, W. Kuo, and Y. Cui. Zero-shot detection via vision and language knowledge distillation. *arXiv:2104.13921*, 2021.

[41] A. Gupta, P. Dollar, and R. Girshick. Lvis: A dataset for large vocabulary instance segmentation. In *CVPR*, 2019.

[42] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[43] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *ICCV*, 2017.

[44] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020.

[45] O. Henaff. Data-efficient image recognition with contrastive predictive coding. In *ICML*, 2020.

[46] O. J. Hénaff, S. Koppula, J.-B. Alayrac, A. v. d. Oord, O. Vinyals, and J. Carreira. Efficient visual pretraining with contrastive detection. *arXiv:2103.10957*, 2021.

[47] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv:1704.04861*, 2017.

[48] G. Huang, H. Berard, A. Touati, G. Gidel, P. Vincent, and S. Lacoste-Julien. Parametric adversarial divergences are good task losses for generative modeling. *arXiv:1708.02511*, 2017.

[49] G. Huang, H. Larochelle, and S. Lacoste-Julien. Are few-shot learning benchmarks too simple? solving them without task supervision at test-time. *arXiv:1902.08605*, 2019.

[50] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. *arXiv:1602.07360*, 2016.

[51] L. Jiao, F. Zhang, F. Liu, S. Yang, L. Li, Z. Feng, and R. Qu. A survey of deep learning-based object detection. *IEEE access*, 7:128837–128868, 2019.

[52] Z. Jie, Y. Wei, X. Jin, J. Feng, and W. Liu. Deep self-taught learning for weakly supervised object localization. In *CVPR*, 2017.

[53] L. Jing and Y. Tian. Self-supervised visual feature learning with deep neural networks: A survey. *T-PAMI*, 2020.

[54] B. Kang, Z. Liu, X. Wang, F. Yu, J. Feng, and T. Darrell. Few-shot object detection via feature reweighting. In *ICCV*, 2019.

[55] L. Karlinsky, J. Shtok, S. Harary, E. Schwartz, A. Aides, R. Feris, R. Giryes, and A. M. Bronstein. Repmet: Representative-based metric learning for classification and few-shot object detection. In *CVPR*, 2019.

[56] S. Khan, M. Naseer, M. Hayat, S. W. Zamir, F. S. Khan, and M. Shah. Transformers in vision: A survey. *arXiv:2101.01169*, 2021.

[57] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *ICLR*, 2014.

[58] A. Lacoste, P. Rodriguez, F. Branchaud-Charron, P. Atighehchian, M. Caccia, I. H. Laradji, A. Drouin, M. Craddock, L. Charlin, and D. V'azquez. Synbols: Probing learning algorithms with synthetic datasets. In *NeurIPS*, 2020.

[59] B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

[60] I. Laradji, P. Rodriguez, O. Manas, K. Lensink, M. Law, L. Kurzman, W. Parker, D. Vazquez, and D. Nowrouzezahrai. A weakly supervised consistency-based learning method for covid-19 segmentation in ct images. In *WACV*, 2021.

[61] I. H. Laradji, N. Rostamzadeh, P. O. Pinheiro, D. Vázquez, and M. Schmidt. Instance segmentation with point supervision. *arXiv:1906.06392*, 2019.

[62] I. H. Laradji, D. Vazquez, and M. Schmidt. Where are the masks: Instance segmentation with image-level supervision. *arXiv:1907.01430*, 2019.

[63] I. H. Laradji, R. Pardinas, P. Rodriguez, and D. Vazquez. Looc: Localize overlapping objects with

count supervision. In *ICIP*, 2020.

[64] I. H. Laradji, N. Rostamzadeh, P. O. Pinheiro, D. Vazquez, and M. Schmidt. Proposal-based instance segmentation with point supervision. In *ICIP*, 2020.

[65] I. H. Laradji, A. Saleh, P. Rodriguez, D. Nowrouzezahrai, M. R. Azghadi, and D. Vazquez. Weakly supervised underwater fish segmentation using affinity lcfcn. *Scientific Reports*, 11(1):1–10, 2021.

[66] A. Li and Z. Li. Transformation invariant few-shot object detection. In *CVPR*, 2021.

[67] B. Li, B. Yang, C. Liu, F. Liu, R. Ji, and Q. Ye. Beyond max-margin: Class margin equilibrium for few-shot object detection. In *CVPR*, 2021.

[68] C. Li, J. Yang, P. Zhang, M. Gao, B. Xiao, X. Dai, L. Yuan, and J. Gao. Efficient self-supervised vision transformers for representation learning. *arXiv:2106.09785*, 2021.

[69] D. Li, J.-B. Huang, Y. Li, S. Wang, and M.-H. Yang. Weakly supervised object localization with progressive domain adaptation. In *CVPR*, 2016.

[70] X. Li, L. Zhang, Y. P. Chen, Y.-W. Tai, and C.-K. Tang. One-shot object detection without fine-tuning. *arXiv:2005.03819*, 2020.

[71] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014.

[72] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017.

[73] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. In *ICCV*, 2017.

[74] S. Liu, T. Li, W. Chen, and H. Li. Soft rasterizer: A differentiable renderer for image-based 3d reasoning. In *ICCV*, 2019.

[75] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *ECCV*, 2016.

[76] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *arXiv:2103.14030*, 2021.

[77] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015.

[78] O. Mañas, A. Lacoste, X. Giro-i Nieto, D. Vazquez, and P. Rodriguez. Seasonal contrast: Unsupervised pre-training from uncurated remote sensing data. *ICCV*, 2021.

[79] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv:1301.3781*, 2013.

[80] J. Munkres. Algorithms for the assignment and transportation problems. *Journal of the society for industrial and applied mathematics*, 5(1):32–38, 1957.

[81] A. Newell and J. Deng. How useful is self-supervised pretraining for visual tasks? In *CVPR*, 2020.

[82] M. Noroozi and P. Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *ECCV*, 2016.

[83] A. v. d. Oord, Y. Li, and O. Vinyals. Representation learning with contrastive predictive coding. *arXiv:1807.03748*, 2018.

[84] B. N. Oreshkin, P. Rodriguez, and A. Lacoste. Tadam: task dependent adaptive metric for improved few-shot learning. In *NeurIPS*, 2018.

[85] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros. Context encoders: Feature learning by inpainting. In *CVPR*, 2016.

[86] P. O. Pinheiro, A. Almahairi, R. Y. Benmalek, F. Golemo, and A. C. Courville. Unsupervised learning of dense visual representations. In *NeurIPS*, 2020.

[87] L. Qiao, Y. Zhao, Z. Li, X. Qiu, J. Wu, and C. Zhang. Defrcn: Decoupled faster r-cnn for few-shot object detection. In *ICCV*, 2021.

[88] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever. Improving language understanding by generative pre-training. 2018.

[89] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. *arXiv:2103.00020*, 2021.

[90] S. Rahman, S. Khan, and N. Barnes. Improved visual-semantic alignment for zero-shot object detection. In *AAAI*, 2020.

[91] K. Rakelly, E. Shelhamer, T. Darrell, A. Efros, and S. Levine. Conditional networks for few-shot semantic segmentation. In *ICLR*, 2018.

[92] S. Ravi and H. Larochelle. Optimization as a model for few-shot learning. In *ICLR*, 2016.

[93] J. Redmon and A. Farhadi. Yolov3: An incremental improvement. *arXiv:1804.02767*, 2018.

[94] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *CVPR*, 2016.

[95] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *NeurIPS*, 2015.

[96] P. Rodriguez, I. Laradji, A. Drouin, and A. Lacoste. Embedding propagation: Smoother manifold for few-shot classification. In *ECCV*. Springer, 2020.

[97] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015.

[98] M. Siam, N. Doraiswamy, B. N. Oreshkin, H. Yao, and M. Jagersand. Weakly supervised few-shot object segmentation using co-attention with visual and semantic embeddings. In *IJCAI*, pages 860–867, 7 2020.

[99] J. Snell, K. Swersky, and R. Zemel. Prototypical networks for few-shot learning. In *NeurIPS*, 2017.

[100] Z. Sun, S. Cao, Y. Yang, and K. Kitani. Rethinking transformer-based set prediction for object detection. *arXiv:2011.10881*, 2020.

[101] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. Torr, and T. M. Hospedales. Learning to compare: Relation network for few-shot learning. In *CVPR*, 2018.

[102] P. Tang, X. Wang, X. Bai, and W. Liu. Multiple instance detection network with online instance classifier refinement. In *CVPR*, 2017.

[103] P. Tang, X. Wang, S. Bai, W. Shen, X. Bai, W. Liu, and A. Yuille. Pcl: Proposal cluster learning for weakly supervised object detection. *T-PAMI*, 42(1):176–191, 2018.

[104] A. Tarvainen and H. Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. *arXiv:1703.01780*, 2017.
[105] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders. Selective search for object recognition. *IJCV*, 104(2):154–171, 2013.
[106] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *NeurIPS*, 2017.
[107] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *ICML*, 2008.
[108] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra, et al. Matching networks for one shot learning. *NeurIPS*, 2016.
[109] F. Wan, C. Liu, W. Ke, X. Ji, J. Jiao, and Q. Ye. C-mil: Continuation multiple instance learning for weakly supervised object detection. In *CVPR*, 2019.
[110] X. Wang, T. Huang, J. Gonzalez, T. Darrell, and F. Yu. Frustratingly simple few-shot object detection. In *ICML*, 2020.
[111] X. Wang, R. Zhang, C. Shen, T. Kong, and L. Li. Dense contrastive learning for self-supervised visual pre-training. In *CVPR*, 2021.
[112] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni. Generalizing from a few examples: A survey on few-shot learning. *ACM Computing Surveys (CSUR)*, 53(3):1–34, 2020.
[113] Y.-X. Wang, D. Ramanan, and M. Hebert. Meta-learning to detect rare objects. In *CVPR*, 2019.
[114] F. Wei, Y. Gao, Z. Wu, H. Hu, and S. Lin. Aligning pretraining for detection via object-level contrastive learning. *arXiv:2106.02637*, 2021.
[115] J. Wu, S. Liu, D. Huang, and Y. Wang. Multi-scale positive sample refinement for few-shot object detection. In *ECCV*, 2020.
[116] X. Wu, D. Sahoo, and S. Hoi. Meta-rcnn: Meta learning for few-shot object detection. In *ACMMM*, 2020.
[117] T. Xiao, X. Wang, A. A. Efros, and T. Darrell. What should not be contrastive in contrastive learning. In *ICLR*, 2020.
[118] T. Xiao, C. J. Reed, X. Wang, K. Keutzer, and T. Darrell. Region similarity representation learning. *arXiv:2103.12902*, 2021.
[119] Y. Xiao and R. Marlet. Few-shot object detection and viewpoint estimation for objects in the wild. In *ECCV*, 2020.
[120] J. Xie, R. Girshick, and A. Farhadi. Unsupervised deep embedding for clustering analysis. In *ICML*, 2016.
[121] X. Yan, Z. Chen, A. Xu, X. Wang, X. Liang, and L. Lin. Meta r-cnn: Towards general solver for instance-level low-shot learning. In *CVPR*, 2019.
[122] C. Yang, Z. Wu, B. Zhou, and S. Lin. Instance localization for self-supervised detection pretraining. In *CVPR*, 2021.
[123] J. Yang, D. Parikh, and D. Batra. Joint unsupervised learning of deep representations and image clusters. In *CVPR*, 2016.
[124] S. S. A. Zaidi, M. S. Ansari, A. Aslam, N. Kanwal, M. Asghar, and B. Lee. A survey of modern deep learning based object detection models. *arXiv:2104.11892*, 2021.
[125] X. Zhai, A. Kolesnikov, N. Houlsby, and L. Beyer. Scaling vision transformers. *arXiv:2106.04560*, 2021.
[126] G. Zhang, Z. Luo, K. Cui, and S. Lu. Meta-detr: Few-shot object detection via unified image-level meta-learning. *arXiv:2103.11731*, 2021.
[127] R. Zhang, P. Isola, and A. A. Efros. Colorful image colorization. In *ECCV*, 2016.
[128] R. Zhang, P. Isola, and A. A. Efros. Split-brain autoencoders: Unsupervised learning by cross-channel prediction. In *CVPR*, 2017.
[129] Y. Zheng, R. Huang, C. Han, X. Huang, and L. Cui. Background learnable cascade for zero-shot object detection. In *ACCV*, 2020.
[130] Y. Zheng, J. Wu, Y. Qin, F. Zhang, and L. Cui. Zero-shot instance segmentation. In *CVPR*, 2021.
[131] X. Zhou, D. Wang, and P. Krähenbühl. Objects as points. *arXiv:1904.07850*, 2019.
[132] X. Zhu, W. Su, L. Lu, B. Li, X. Wang, and J. Dai. Deformable detr: Deformable transformers for end-to-end object detection. In *ICLR*, 2021.

Gabriel Huang is a PhD candidate at Mila, Québec Institute for Learning Algorithms, and DIRO, Université de Montréal. He is also a research intern at Element AI, a ServiceNow company. His research focuses on few-shot learning, self-supervised learning, and learning structured representations.

Issam Laradji is a research scientist at ServiceNow in the low data learning lab. He completed his PostDoc at the McGill graphics lab led by Derek Nowrouzezahrai. His main research interests are in differentiable rendering and physics, weakly supervised learning and optimization for computer vision and natural language processing applications.

David Vazquez is a Research Lead at ServiceNow working on machine learning methods that can efficiently train with few supervision applied to computer vision, and natural language processing tasks. He finished a postdoc at Montreal Institute of Learning Algorithms (Mila) and a PhD at Universitat Autonoma de Barcelona (UAB).

Simon Lacoste-Julien is the associate director of the CIFAR Learning in Machines & Brains program and Canada CIFAR AI Chair at Mila. He is an associate professor at the Department of Computer Science and Operations Research (DIRO) at Université de Montréal and is the part-time VP Lab Director at the Samsung SAIT AI Lab in Montreal. Simon Lacoste-Julien's research focuses on machine learning, i.e., how to program a computer to learn from data and solve useful tasks.

Pau Rodriguez is a Research Scientist at Element AI, a ServiceNow company, adjunct professor at UAB and ELLIS member. He obtained a PhD in Computer Vision and Artificial Intelligence at UAB.