Hallucination Improves Few-Shot Object Detection

Weilin Zhang Yu-Xiong Wang
University of Illinois at Urbana-Champaign
{weilinz2, yxw}@illinois.edu

Abstract

Learning to detect novel objects from few annotated examples is of great practical importance. A particularly challenging yet common regime occurs when there are extremely limited examples (less than three). One critical factor in improving few-shot detection is to address the lack of variation in training data. We propose to build a better model of variation for novel classes by transferring the shared within-class variation from base classes. To this end, we introduce a hallucinator network that learns to generate additional, useful training examples in the region of interest (RoI) feature space, and incorporate it into a modern object detection model. Our approach yields significant performance improvements on two state-of-the-art few-shot detectors with different proposal generation procedures. In particular, we achieve new state of the art in the extremely-few-shot regime on the challenging COCO benchmark.

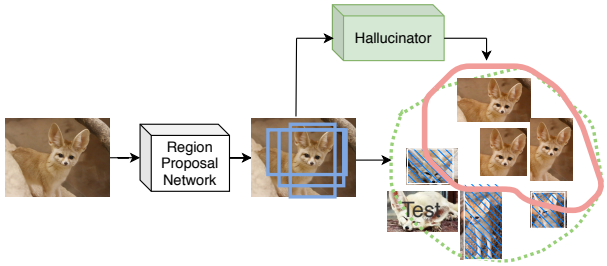

Figure 1. Learning to detect a novel class, *fennec fox*, from a single training example (*i.e.*, 1-shot detection) using a serial detector. The region proposal network (RPN) generates a few high intersection-over-union (IoU) boxes for the detector's classifier. The pink circle represents the classifier decision boundary learned from these boxes. Due to a lack of sample variation, the decision boundary is not accurately estimated. With hallucinated examples (image in backslash) produced by our hallucinator, the classifier learns a better decision boundary (the dotted circle), thus being able to potentially correct previously misclassified instances.

1. Introduction

Modern deep convolutional neural networks (CNNs) rely heavily on large amounts of annotated images [29]. This data-hungry nature limits their applicability to some practical scenarios such as autonomous driving, where the cost of annotating examples is prohibitive, or which involve *never-before-seen* concepts [9,51]. By contrast, humans can rapidly grasp a new concept and make meaningful generalizations, even from a single example [31]. To bridge this gap, there has been a recent resurgence of interest in few-shot or low-shot learning that aims to learn novel concepts from very few labeled examples [8, 10, 34, 37, 42].

Despite notable successes, most of the existing work has focused on simple classification tasks with artificial settings and small-scale datasets [34, 37]. However, few-shot object detection, a task of great practical importance that learns an object detector from only a few annotated bounding box examples [18, 38, 39], is far less explored. Few-shot detection requires determining *where* an object is as well as *what* it is (and handling distracting background regions [13], *etc.*), and is much harder than few-shot classification. The most difficult regime occurs when there are *very limited examples* (less than 3) for novel classes (Figure 1), which is a common yet extremely challenging case in the real world.

While few-shot classification approaches are helpful (*e.g.*, [2, 4, 18, 33, 41]), few-shot detection is much more than a straightforward application of few-shot classification approaches. The state-of-the-art two-stage fine-tuning approach (TFA) [38] learns a better representation for few-shot detection, through (1) pre-training on base classes with abundant data and then only fine-tuning the box classifier and regressor on novel classes, and (2) introducing instance-level feature normalization to the box classifier during fine-tuning. Despite the improvement, its performance in the extremely low-data regime is still far from satisfying.

We argue that, to fully improve *extremely-few-shot* detection performance, a key factor is to effectively deal with *the lack of variation in training data*. This is because for an object detector to be accurate, its classifier must build a useful model of variation in appearance with very few examples. More concretely, a modern object detector first finds promising image locations, typically boxes, using a region proposal network (RPN) [28], then passes promising boxes

through a classifier to determine what object is present, and finally performs various cleanup operations such as non-maximum suppression (NMS), aimed at avoiding duplicate predictions and improving localization. Now assume that the detector must learn to detect a novel category from a single example (Figure 1). The only way the classifier can build a model of the category's variation in appearance is by learning from the high intersection-over-union (IoU) boxes reported by the RPN. Although there is variation of boxes produced by the RPN, the variation from a single example is too weak to train the classifier for the novel class.

To overcome this issue, one strategy is to adjust the learning procedure for RPN, so that it reports highly informative boxes. Contemporary work [48] achieves this by training multiple RPN's be somewhat redundant and cooperating. Hence, if one RPN misses a highly informative box, another will get it. This cooperating RPN's (CoRPNs) approach, while helpful, is still insufficient. In the extremely-few-shot regime, all positive novel class proposals produced by the multiple RPN's are only slightly modified from and thus similar to the few available positive instances (with light-weighted cropping and scaling operations); their variation is significantly limited for building a strong classifier.

In this paper, we propose a different perspective on building a model of variation for novel classes, *by transferring the shared within-class variation from base classes*. In fact, many modes of variation in the visual world (*e.g.*, camera pose, lighting changes, and even articulation) are shared across categories and can generalize to unseen classes [30]. While such within-class variation is difficult to be encoded through the proposal generation procedure, it can be effectively captured by *learning to hallucinate examples* [40].

To this end, we introduce a *hallucinator network* into a modern object detection model. The hallucinator network performs data hallucination for the box classifier in the learned region of interest (RoI) feature space. We train the hallucinator on data-abundant base classes, encoding the rich structure of their shared modes of variation. We then use the learned hallucinator to generate additional novel class examples and thus produce an augmented training set for building better classifiers, as shown in Figure 1.

Note that the existing strategy for training the hallucinator in few-shot classification [40] is coupled with a complicated meta-learning process, making it difficult to apply to state-of-the-art few-shot detectors like TFA [38] or CoRPNs [48]. We overcome this challenge by introducing a much simpler yet effective training procedure: we train our hallucinator and the detector's classifier in an *EM-like* (expectation-maximization) manner, where a "strongest possible" classifier is trained first with all the available base class data; the hallucinator is then trained under the guidance of this already-trained classifier; and finally, the classifier is re-trained and refined based on the set of augmented examples (with hallucinated examples) on novel classes.

Our contributions are three-fold. (1) We investigate a critical yet under-explored issue in extremely-few-shot detection (*e.g.*, as few as one) – the lack of variation in training data. (2) We propose a novel data hallucination based approach to address this issue, which effectively transfers shared modes of within-class variation from base classes to novel classes. Our approach is simple, general, and can work with different region proposal procedures. (3) Our approach significantly outperforms the state-of-the-art TFA [38] and most recent cooperating RPN's [48] detectors in the extremely-few-shot regime. Our code is available at https://github.com/pppplin/HallucFsDet.

2. Related Work

Object Detection: Modern detectors are typically based on convolutional neural networks with two types of architectures – serial detection [12] and parallel detection [25]. Both families run a region proposal process [5, 36] that determines whether an image region contains an object or not. They differ in when to run the region proposal process. Serial detectors (or two-stage detectors) first generate promising region proposals and then feed each proposal box to a classifier that predicts if the region contains an object. Serial detectors include R-CNN [13] and its variants, such as Fast R-CNN [12], Faster R-CNN [28], Mask R-CNN [15], SPP-Net [16], FPN [21], and DCN [3].

Parallel detectors (or one-stage detectors) run the region proposal process and the classification process simultaneously. Parallel detectors are usually faster than serial detectors at the expense of decreasing accuracy, since the classifier has no prior knowledge of whether a box contains an object. Representative parallel detectors are variants of YOLO [1, 25, 26, 27], SSD [23], CornerNet [19], and ExtremeNet [49]. Our work introduces a general data hallucination strategy to improve the detection performance in the few-shot regime. While we mainly focus on serial detectors in this paper since they achieve state-of-the-art few-shot detection performance, we believe that our strategy applies to parallel detectors as well, which we leave as future work.

Few-Shot Object Detection: Advanced few-shot detectors are usually built in a serial fashion [7, 38, 39, 43, 44, 45]. One line of work focuses on learning better feature representations through metric learning [20, 33, 46], in ways of modeling multi-modal distributions in each class [33], restoring negative information [46], and reserving adequate margin space among novel classes [20]. Modified fine-tuning techniques have also been explored. For example, a regularized fine-tuning approach is proposed to transfer knowledge from a pre-trained detector to a few-shot detector [2]. Recently, a simple two-stage fine-tuning approach has been shown to outperform more sophisticated methods [38]. Other work seeks improvements by applying

meta-learning techniques, such as learning a meta-model to reweight pre-trained features given few-shot data [18] and attaching meta-learned classifiers to Faster R-CNN [39,45].

Another line of work focuses on improving the proposal generation process, by introducing attention mechanisms and generating class-aware features for classifiers [7, 17, 24, 35, 44, 48]. Some work modifies the proposal ranking process and ranks proposals based on similarity with query images [7,17]. An RPN ensemble method is proposed to avoid missing highly informative proposal boxes [48]. Contrastive-aware object proposal encodings are further learned to reduce the possibility of misclassifying novel class objects to confusable classes [35]. Additional information has also been shown helpful, such as semantic relations [50] and multi-scale representations [43]. *Orthogonal to existing work*, we address few-shot detection by hallucinating additional data and enriching sample variation.

Data Hallucination: Despite recent progress on learning to hallucinate examples to deal with data scarcity [11, 14, 32, 40, 47], much of the work has focused on classification tasks and performed in a learned feature space. Novel class features are generated by learning shared feature transformations from base classes [14]. Pairwise deformations between examples of the same class are captured and used to generate novel class instances [32]. A meta-learner and a hallucinator are combined and jointly optimized to boost recognition performance [40]. Our approach builds on the feature space hallucination framework of [40], but makes significant modifications to the hallucinator architecture and the hallucination procedure. To the best of our knowledge, our work is the *first* to demonstrate the effectiveness of data hallucination for few-shot detection.

3. Our Approach

We believe that hallucination would improve any few-shot detection models with standard architectures. In this section, we focus on improving two state-of-the-art few-shot detectors with *different region proposal mechanisms*.

Few-Shot Object Detection Setting and Evaluation Procedure: We follow the few-shot detection setting and the standard evaluation procedure introduced in [18, 38]. Classes are split into two sets: base classes C_b and novel classes C_n. Both models which we study adopt a two-stage fine-tuning procedure [38]. A model is trained on base classes in the first stage and is then fine-tuned on novel classes in the second stage. In the first stage, the model only detects base classes – the model is a $|C_\text{b}|$-way detector. In the second stage, the model is expanded as a $(|C_\text{b}| + |C_\text{n}|)$-way detector. In order to maintain performance on base classes while learning to detect novel class instances, the detector is fine-tuned on a *balanced* few-shot dataset containing both novel and base classes. Finally, we compare mean average precision (AP) on novel classes.

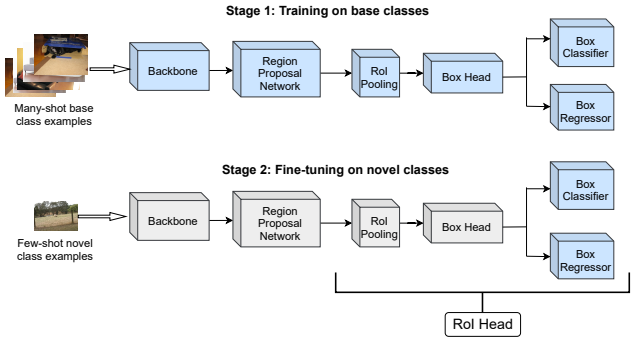

Figure 2. Illustration of the two-stage fine-tuning approach (TFA) [38] under the Faster R-CNN framework. In stage 1, the whole model is trained with base class images. In stage 2, only the classifier and the bounding box regressor in the RoI head are fine-tuned with novel class images. A module in blue is trained during the corresponding stage, while a module in gray is frozen.

3.1. Few-Shot Object Detection Models

Two-stage Fine-tuning Approach (TFA): A two-stage fine-tuning approach is introduced [38] under the widely-used Faster R-CNN framework, which significantly improves few-shot detection performance. As a serial detector, Faster R-CNN consists of a backbone, a region proposal network (RPN), a region of interest (RoI) pooling layer, an RoI feature extractor, a bounding box classifier, and a bounding box regressor (Figure 2). The last four components together construct the RoI head. The backbone extracts image features, which are passed through the RPN to produce promising areas with potential objects. The RoI head then transforms, classifies, and refines potential object boxes into labeled boxes. TFA modifies the standard Faster R-CNN by using a cosine similarity based classifier to reduce intra-class variance for few-shot learning. TFA uses an ImageNet [29] pre-trained ResNet-101 with a feature pyramid network [21] as the backbone. As shown in Figure 2, in the first stage, TFA trains the whole model using base class images. In the second stage, only the classifier and the bounding box regressor in the RoI head are fine-tuned using novel class instances, with the rest of the model frozen.

Cooperating RPN's (CoRPNs): An improved proposal generation mechanism based on RPN ensemble is proposed in CoRPNs [48], which produces highly informative proposal boxes for few-shot detection. As shown in Figure 3, CoRPNs train multiple distinct but cooperating RPN's so that if one RPN misses a highly informative proposal box, another one will get it. Except the proposal generation procedure, CoRPNs share the same model architecture and training strategy with TFA. Specifically, CoRPNs train N RPN's by a modified RPN classification loss

$$\mathcal{L} = \mathcal{L}_\text{CE}^{j^*} + \mathcal{L}_\text{div} + \mathcal{L}_\text{coop}, \tag{1}$$

where $\mathcal{L}_\text{CE}^{j^*}$ is a binary cross-entropy loss of a selected RPN

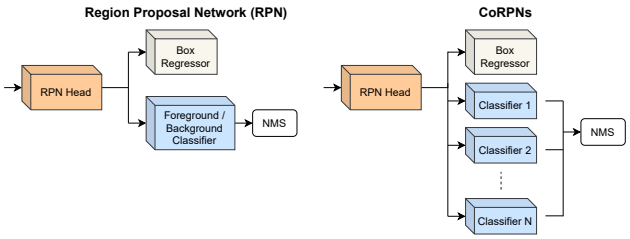

Figure 3. Illustration of cooperating RPN's (CoRPNs) [48]. **Left**: the standard RPN structure in Faster R-CNN. **Right**: CoRPNs with N RPN classifiers.

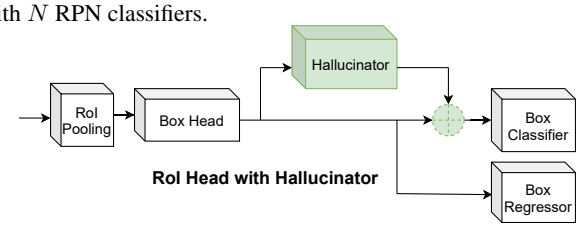

Figure 4. Illustration of the RoI head structure with a hallucinator module. The hallucinator is inserted right before the classifier. The hallucinated examples are appended to the original training examples to train the classifier. The bounding box regressor is not affected by the hallucinator.

j^*. CoRPNs select the most certain RPN for every box. For instance, each anchor box will get the RPN j^*'s score that has a probability closer to either 0 or 1. The selected RPN j^* gets this box's gradient at training.

The divergence loss $\mathcal{L}_{\text{div}}$ encourages RPN's to be different, and the cooperation loss $\mathcal{L}_{\text{coop}}$ enforces RPN's to cooperate by setting a lower bound for every RPN's response to a foreground box. $\mathcal{L}_{\text{div}}$ is defined as

$$\mathcal{L}_{\text{div}} = -\log(\det(\Sigma(\mathcal{F}))), \tag{2}$$

which is the negative log of the covariance matrix on every RPN's prediction to proposal boxes. For foreground box i and RPN j, the cooperation loss is

$$\mathcal{L}^{i,j}_{\text{coop}} = \max\{0, \phi - f^j_i\}, \tag{3}$$

where ϕ is the lower bound of every RPN's response to a foreground box, and f^j_i is RPN j's response to the box i. $\mathcal{L}_{\text{coop}}$ is averaged over all RPN's and foreground boxes.

3.2. Few-Shot Object Detection with Hallucination

We introduce a hallucinator network H with parameters ϕ that learns to generate additional examples for novel classes by leveraging the shared within-class feature variation from base classes. As shown in Figure 4, hallucination happens in the RoI head feature space. The hallucinator takes as input available training examples and generates hallucinated examples. The set of hallucinated examples S_{gen} is then treated as additional training examples for learning the classifier on novel classes. Specially, given a seed example $x^{c_k}_i$ of category c_k, the hallucinator generates a hallucinated example by

$$\widetilde{x^{c_k}_i} = H(\mu_{c_k}, x^{c_k}_i, \varepsilon; \phi), \tag{4}$$

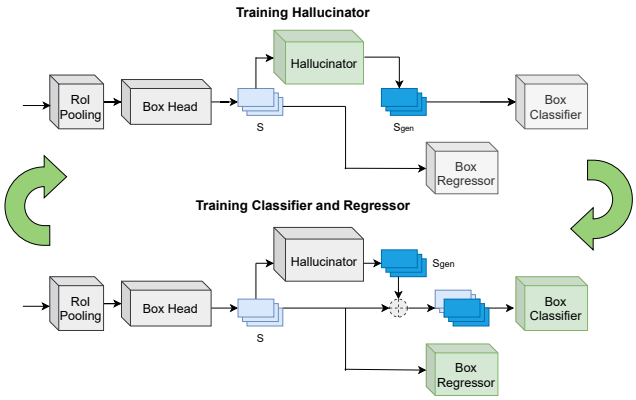

Figure 5. Our EM-style training procedure for an RoI head with a hallucinator module. **Upper:** how to train the hallucinator. The hallucinator is trained based on the classification loss on the hallucinated examples. All other modules are frozen when training the hallucinator. **Lower**: how to train the classifier and the bounding box regressor. Both the original training examples and the hallucinated examples are used to train the classifier. Only the original training examples are used to train the bounding box regressor. The EM-style training *iterates* between the two processes. A gray module indicates that it is frozen.

where μ_{c_k} is the class prototype of c_k, which is the mean of all instances from c_k, and ε is a per-example noise vector. Note that our hallucinator formulation is different from [40]: we introduce an additional input μ_{c_k} to explicitly capture the *global* category information. For $\widetilde{x^{c_k}_i}$, its label y_i is the same as the seed example's category c_k. Assuming that we already have a trained box classifier, we directly use the classifier's (cross-entropy) classification loss on *all hallucinated examples* to train the hallucinator:

$$L_{\text{halluc}}(\phi) = \sum_{i \in S_{\text{gen}}} \left(-w^T_{y_i} \widetilde{x^{c_k}_i} + \log \sum_k e^{w^T_k \widetilde{x^{c_k}_i}} \right), \tag{5}$$

where w_{y_i} and w_k are the *already-learned* classifier's weights on corresponding categories. This hallucination loss enforces the hallucinated examples to loosely "agree with" the trained classifier. In addition, since we take as input class prototypes, our hallucinator is implicitly regularized and thus not simply copying the seed examples. Compared with [40] which uses a complicated meta-learning process, we substantially simplify the training procedure.

3.3. Training the Hallucinator and the Classifier

As illustrated in Figure 5, we propose an EM-style (expectation–maximization) training procedure to train the hallucinator and the detector's classifier: when training the hallucinator, the classifier is frozen; when training the classifier (and the box regressor), the hallucinator is frozen. Compared with end-to-end joint training, this iterative strategy improves the cooperation between the hallucinator and the classifier, thus producing examples that are more useful for building the few-shot classifier.

Training on Base Classes: In the stage of training on base classes, we first train the plain detector *without the hallucinator* in a standard way as before. We then insert the hallucinator into the detector, and freeze all model components except the hallucinator. Now, the hallucination will be performed in the pre-trained, fixed RoI head feature space. And the classifier is already trained on base classes using all available examples. We then use this base class classifier to guide the training of the hallucinator.

Note that, different from few-shot classification, the proposal generation process in detection produces several proposal boxes around an object, making a few training examples available even in the 1-shot scenario. We randomly sample these training examples as seed examples for the hallucinator. Consider a training batch S which consists of n base classes. For each class in the batch, the hallucinator generates m examples, and m is fixed for all batches. The set of hallucinated examples S_{gen} is used to train the hallucinator based on the loss function (5).

Fine-Tuning on Novel Classes: After training the hallucinator on base classes, we move on to the stage of fine-tuning the classifier and hallucinator on a few-shot dataset containing both base and novel classes. We freeze all model components prior to the hallucinator. Each training batch S initially consists of an imbalanced set of foreground box examples S_{pos} and background box examples S_{neg}, with background examples being the majority: $S = [S_{\text{pos}}; S_{\text{neg}}]$. The hallucinator generates a set of additional examples S_{gen} for novel classes. *Without changing the number of examples in total*, we randomly replace $|S_{\text{gen}}|$ background examples by S_{gen} to obtain a refined training batch

$$\widetilde{S} = [S_{\text{pos}}, S_{\text{gen}}; S'_{\text{neg}}]. \tag{6}$$

By doing so, we also partially alleviate the imbalance issue between foreground and background examples. After training the classifier on the refined dataset with hallucinated examples, we fine-tune the hallucinator using the classifier based on Eq. (5), then we use the fine-tuned hallucinator to fine-tune the classifier again, and so on. We stop this procedure after one or two iterations (*i.e.*, the classifier is fine-tuned at most twice); empirically, we found that additional iterations do not further improve the performance.

4. Experiments

4.1. Implementation Details

Datasets and Evaluation Metrics: We evaluate on two widely-used few-shot detection benchmarks: MS-COCO [22] and PASCAL VOC (07 + 12) [6]. For a fair comparison, the *same* base/novel category splits, train/test splits, and novel class instances [18, 38] are used for training and evaluation. Consistent with recent work [18,38], we report AP50 for three different base/novel splits under shots 1, 2, 3, 5, and 10 on PASCAL VOC; we report AP, AP50, and AP75 under shots 1, 2, and 3 on COCO. Note that we *particularly focus on the extremely-few-shot regime*.

Baselines: We mainly focus on the comparison with two state-of-the-art few-shot detectors – TFA [38] and CoRPNs [48], and evaluate the effectiveness and generalizability of our hallucinator when combined with them. For both our models and the main baselines, we use Faster R-CNN [28] as our base model. Following [38], we use an ImageNet pre-trained ResNet-101 with a feature pyramid network [21] as the backbone. All our experiments, including those with the main baselines, have ground-truth boxes appended as training examples in the RoI head. As reported in [38], including ground-truth boxes leads to a 0.5% AP gain on COCO. In addition, we compare with a variety of baselines, including concurrent work on few-shot detection.

Evaluation Procedure: Both our models and the main baselines are $(|C_{\text{b}}| + |C_{\text{n}}|)$-way few-shot detectors. And we evaluate them on both base and novel classes under the standard evaluation procedure [38], as described in Section 3. Some other baselines like [17] were initially evaluated under different procedures. For a fair comparison, *all reported numbers for those methods are the re-evaluated results under the standard evaluation procedure*.

Note that, following [38], the fine-tuning stages on PASCAL VOC and COCO are slightly different. On PASCAL VOC, the classifier's weights on novel classes are randomly initialized and directly trained using a balanced dataset with base and novel classes. On COCO, we first train a $|C_{\text{n}}|$-way classifier on novel class instances. The trained classifier is used as an initialization of the classifier's weights on novel classes. We then train a $(|C_{\text{b}}| + |C_{\text{n}}|)$-way classifier using a balanced few-shot dataset with base and novel classes.

Hallucinator Architecture: We use a two-layer MLP (multi-layer perceptron) with ReLU as the hallucinator. Given that the inputs to the hallucinator are a class prototype, a seed example, and random noise, the input size is three times the feature size; the output size of each linear layer is the same size as the input feature. The seed example is randomly sampled from all examples in a training batch. The input noise is dataset-dependent. For each dataset, we calculate the mean and standard deviation of pre-trained input features and use those to sample the input noise.

We pre-compute base class prototypes using all available examples before training the hallucinator. During training the hallucinator, we do not update these base class prototypes. In a different manner, we construct novel class prototypes *dynamically*, when using the learned hallucinator to train classifiers on novel classes. Specifically, novel class prototypes are computed by using both the training and hallucinated examples (in contrast to that only training examples are used for base class prototypes). For a novel class, whenever a new training example comes in or a new example is generated, its corresponding prototype will be up-

		Novel Set 1					Novel Set 2					Novel Set 3				
	Method	shot=1	2	3	5	10	shot=1	2	3	5	10	shot=1	2	3	5	10
Ours	CoRPNs w/ cos + Halluc	**47.0**	**44.9**	**46.5**	**54.7**	54.7	**26.3**	**31.8**	**37.4**	**37.4**	41.2	**40.4**	**42.1**	43.3	51.4	49.6
Main baseline	CoRPNs w/ cos [48]	44.4	38.5	46.4	54.1	**55.7**	25.7	29.5	37.3	36.2	**41.3**	35.8	41.8	**44.6**	**51.6**	49.6
Ours	TFA w/ cos + Halluc	**45.1**	**44.0**	44.7	55.0	55.9	23.2	**27.5**	**35.1**	34.9	39.0	30.5	**35.1**	41.4	49.0	49.3
Main baseline	TFA w/ cos [38]	39.8	36.1	44.7	**55.7**	**56.0**	**23.5**	26.9	34.1	**35.1**	**39.1**	**30.8**	34.8	**42.8**	**49.5**	**49.8**
	FRCN+ft-full [38]	15.2	20.3	29.0	40.1	45.5	13.4	20.6	28.6	32.4	38.8	19.6	20.8	28.7	42.2	42.1
	Meta R-CNN [45]	19.9	25.5	35.0	45.7	51.5	10.4	19.4	29.6	34.8	45.4	14.3	18.2	27.5	41.2	48.1
	CoAE* [17]	12.7	14.6	14.8	18.2	21.7	4.4	11.3	20.5	18.0	19.0	6.3	7.6	9.5	15.0	19.0
	MPSR [43]	41.7	43.1	51.4	55.2	61.8	24.4	29.5	39.2	39.9	47.8	35.6	40.6	42.3	48.0	49.7
Other baselines	FsDetView [44]	24.2	35.3	42.2	49.1	57.4	21.6	24.6	31.9	37.0	45.7	21.2	30.0	37.2	43.8	49.6
	NP-RepMet [46]	37.8	40.3	41.7	47.3	49.4	41.6	43.0	43.4	47.4	49.1	33.3	38.0	39.8	41.5	44.8
	FSCE [35]	44.2	43.8	51.4	61.9	63.4	27.3	29.5	43.5	44.2	50.2	37.2	41.9	47.5	54.6	58.5
	CME [20]	41.5	47.5	50.4	58.2	60.9	27.2	30.2	41.4	42.5	46.8	34.3	39.6	45.1	48.3	51.5
	SRR-FSD [50]	47.8	50.5	51.3	55.2	56.8	32.5	35.3	39.1	40.8	43.8	40.1	41.5	44.3	46.9	46.4

Table 1. Few-shot detection performance (AP50) on PASCAL VOC novel classes under three base/novel splits. We follow the standard evaluation procedure in [38]. *The main comparison* focuses on combining hallucination (denoted as 'Halluc') with two state-of-the-art few-shot detectors: TFA and CoRPNs with cosine classifier (denoted as 'w/ cos'). All models build upon Faster R-CNN with ResNet-101 backbone. 'CoRPNs + Halluc' and CoRPNs *share the same hyper-parameter settings*; and the RPN outputs in 'CoRPNs + Halluc' are the same as CoRPNs. Similarly, 'TFA + Halluc' and TFA share the same RPN outputs. Hallucination yields significant improvements in the extremely-few-shot regime (1-shot and 2-shot). In higher shots, hallucination maintains comparable results to its baselines. In addition, we compare with other state-of-the-art methods, including some concurrent work. *Model re-evaluated under the standard procedure. Results in red are the best, results in blue are the second-best, and results in **bold** are the better ones in comparison with the corresponding baseline.

dated. Despite somewhat inconsistency between base and novel classes for constructing prototypes, we found that this strategy exploits all available examples and empirically achieves the best performance.

Hallucinator Initialization: Following a similar strategy as in [40], we initialize the hallucinator by using block diagonal identity matrices plus small random noise. The initialization noise is sampled from a normal distribution with a zero mean and a standard deviation of 0.02. As a form of regularization, this identity initialization ensures that the initially hallucinated examples are not too far away from the seed examples and thus do not degrade the performance.

Hyper-Parameter Settings: On PASCAL VOC, we train the hallucinator with batch size 16 and learning rate 0.02 for 8,000 iterations. We decay the learning rate by ratio 0.1 at 2,000 and 6,000 iterations. On COCO, we train the hallucinator with batch size 64 and learning rate 0.02 for 21,200 iterations. We decay the learning rate by ratio 0.1 at 6,400 and 19,200 iterations.

Number of Hallucinated Examples: When training the classifier on novel classes, the hallucinator produces a fixed number (m) of examples per category in each training batch. For simplicity, we set m to be the same as the average number of per-class training examples (*i.e.*, RoI features), averaged over the training batches. On PASCAL VOC, there are roughly 20 training examples per class (with batch size 16), and we thus hallucinate 20 examples per class accordingly *for all experiments*. We keep this number on COCO as well. In the ablation study, we show that by tuning the number of hallucinated examples, the performance gets further improved. Note that this number should change when the number of images per batch varies.

4.2. Main Results

Comparison with Main Baselines TFA and CoRPNs: Tables 1 and 2 summarize the results for novel classes on PASCAL VOC and COCO, respectively. We have the following key observations.

(1) Hallucination improves the performance over the main baselines *by large margins in the extremely-few-shot regime*. On PASCAL VOC, hallucination yields substantial improvements in 1-shot and 2-shot scenarios. On the more challenging COCO benchmark, hallucination *consistently* improves performance when combined with both baselines.

(2) Our hallucination strategy is *general and applicable to different types of detectors regardless of RPN outputs*. This is because hallucination provides additional, useful sample variation that is independent of RPN outputs.

(3) On PASCAL VOC (Table 1), hallucination demonstrates slightly different behaviors when combined with CoRPNs or TFA: while hallucination consistently improves CoRPNs in the extremely-few-shot regime, it is not always helpful for TFA in some cases (though hallucination also does not hurt). A possible explanation is that our hallucinator tends to be *conservative*, making the hallucinated examples still close to the original seed training examples. Hence, in the cases (*e.g.*, novel sets 2 and 3) where the training and test examples are quite distinct, the hallucinated examples produced from the training examples could not help build a better classifier for detecting the test examples. In such cases, a more *aggressive* hallucinator is desired to generate examples that are further away from the seed ex-

	Method	1-shot AP	1-shot AP50	1-shot AP75	2-shot AP	2-shot AP50	2-shot AP75	3-shot AP	3-shot AP50	3-shot AP75
Ours	CoRPNs w/ cos + Halluc	**4.4**	**7.5**	**4.9**	**5.6**	**9.9**	**5.9**	**7.2**	**13.3**	**7.4**
Main baseline	CoRPNs w/ cos [48]	4.1	7.2	4.4	5.4	9.6	5.6	7.1	13.2	7.2
Ours	TFA w/ cos + Halluc	**3.8**	**6.5**	**4.3**	**5.0**	**9.0**	**5.2**	**6.9**	**12.6**	**7.0**
Main baseline	TFA w/ cos [38]	3.4	5.8	3.8	4.6	8.3	4.8	6.6	12.1	6.5
Other baselines	MPSR** [43]	2.3	4.1	2.3	3.5	6.3	3.4	5.2	9.5	5.1
	FsDetView [44]	3.2	**8.9**	1.4	4.9	**13.3**	2.3	6.7	**18.6**	2.9

Table 2. Few-shot detection performance on COCO novel classes in 1, 2, and 3-shot scenarios. See the caption of Table 1 for details. Hallucination *consistently improves both baselines in all settings*. 'CoRPNs + Halluc' achieves the state-of-the-art results in most cases. **Model evaluated using the public code and the pre-trained detector on base classes.

	Method	1-shot fine-tuned AP	1-shot fine-tuned AP50	1-shot fine-tuned AP75	2-shot fine-tuned AP	2-shot fine-tuned AP50	2-shot fine-tuned AP75	3-shot fine-tuned AP	3-shot fine-tuned AP50	3-shot fine-tuned AP75
Ours	CoRPNs w/ cos + Halluc	32.3	52.4	34.4	34.5	55.3	37.0	34.7	55.1	37.5
Main baseline	CoRPNs w/ cos [48]	34.1	55.1	36.5	34.7	55.3	37.3	34.8	55.2	37.6
Ours	TFA w/ cos + Halluc	31.5	50.8	33.9	32.8	52.4	35.4	33.3	52.8	36.4
Main baseline	TFA w/ cos [38]	34.1	54.7	36.4	34.7	55.1	37.6	34.7	54.8	37.9
Other baselines	MPSR** [43]	12.1	17.1	14.2	14.4	20.7	16.9	15.8	23.3	18.3
	FsDetView [44]	2.4	7.0	1.0	4.4	11.9	2.2	4.9	13.6	2.2

Table 3. Detection performance on COCO *base* classes, *after fine-tuning* with a balanced dataset (1, 2, and 3-shots) containing base and novel class instances. Hallucination slightly degrades base class performance compared with CoRPNs and TFA, but it substantially outperforms state-of-the-art MPSR and FsDetView. **Model evaluated using the public code and the pre-trained detector on base classes.

amples. The ablation study in Section 4.3 investigates this issue in more depth and supports our observation.

(4) The performance gain of hallucination diminishes as the number of training examples increases. In higher-shot scenarios, our current hallucination does not bring in additional within-class variation beyond the given training examples, and is thus no longer beneficial. Note that, however, hallucination does not hurt the performance in 5/10-shot scenarios – performance variation is within 95% confidence interval over multiple random samples according to [38]. Given that a *single* hallucinator is trained in our case, we might need different hallucinators or hallucination strategies in different sample size regimes.

Comparison with Other State-of-the-Art Methods: Tables 1 and 2 show that our approach is superior or comparable to other few-shot detection methods. In particular, we achieve very competitive results in the extremely-few-shot regime. We believe that our effort is orthogonal to concurrent work; our hallucination strategy can be combined with other methods to further improve their performance.

Base Class Performance: Table 3 presents the detection results on base classes, evaluated after fine-tuning with novel class instances. We observe that hallucination improves novel class performance at the expense of *slight* degradation of performance on base classes. This issue could be potentially addressed by using a smaller fine-tuning learning rate for the classifier components of base classes (which have been already well-trained). More notably, compared with other state-of-the-art methods such as MPSR [43] and FsDetView [44], we achieve a significantly better trade-off between novel and base class performance.

Method	Novel Set 1	Novel Set 2	Novel Set 3
Joint training	38.5	22.2	33.6
EM w/ 1 iter	46.7	24.5	38.5
EM w/ 2 iter	**47.0**	**26.3**	**40.4**

Table 4. Comparison (1-shot AP50) of different training procedures for CoRPNs with hallucination on PASCAL VOC. All procedures use *the same hallucinator architecture*. *EM-style training significantly outperforms joint training* under all base/novel splits. The second EM iteration also *consistently improves performance*.

4.3. Ablation Studies

EM vs. Joint Training Procedures: Table 4 shows that, with the same hallucinator architecture, our EM-style training procedure significantly outperforms joint training, indicating that joint training might be greedy in this case. In addition, 2 EM-iterations outperform 1 iteration, and we found that more iterations do not improve the performance.

Conservative vs. Aggressive Hallucinators: In the main experiments, we show that a conservative hallucinator is not always helpful and in some cases, we need a more aggressive hallucinator. Here we build such an aggressive hallucinator by *changing the hallucination space*. As shown in Figure 4, the original hallucinator generates examples *directly in the operational space of the classifier* (*i.e.*, in the feature space after the box head and right before the classifier); this makes it conservative and reluctant to produce the kinds of examples that drastically change the classifier decision boundaries. By contrast, our new hallucinator generates examples in the feature space *before the box head* and accordingly, it becomes a three-layer convo-

Method	Novel Set 1	Novel Set 2	Novel Set 3
TFA [38]	39.8	23.5	30.8
TFA + Halluc, Conservative	**45.1**	23.2	30.5
TFA + Halluc, Aggressive	40.8	**29.3**	**33.7**

Table 5. Analysis of conservative and aggressive hallucinators: 1-shot AP50 on PASCAL VOC under three base/novel splits. The aggressive hallucinator brings substantial improvements over TFA on novel sets 2 and 3 without hurting the performance on novel set 1. The conservative hallucinator significantly outperforms TFA on novel set 1, and performs comparably on the other two novel sets.

Method	#Halluc	AP50
CoRPNs [48]	0	44.4
CoRPNs + Halluc	1	47.1
	2	47.0
	3	**48.0**
	5	47.9
	10	47.8
	20	47.0

Table 6. Impact of the number of hallucinated examples on 1-shot detection (AP50) under PASCAL VOC novel set 1.

lutional network. Now, the box head makes allowances for errors in the hallucination, and the hallucinator becomes more aggressive and allows radical change of classifiers. We also make some additional modifications as follows: (1) we jointly train the aggressive hallucinator with other model components, and (2) we use a cosine prototypical network loss [34] computed on held-out validation examples with hallucinated prototypes as additional guidance to train the hallucinator. Table 5 shows that the aggressive hallucinator significantly helps the cases where the performance of the conservative hallucinator is lagging. Importantly, neither of the two hallucinator variants hurts the performance.

Number of Hallucinated Examples: Our hallucinator generates a fixed number of examples per class in each training batch (20 examples in the main results). Table 6 investigates the impact of the number of hallucinated examples. As it increases from 0 to 20, the performance gradually improves and then saturates and drops slightly. Note that the drop is still within 95% confidence interval over multiple random samples according to [38]. This is because even if we produce a small number of hallucinated examples per training batch (*e.g.*, 3), after many iterations, the *cumulative* number of hallucinated examples is large enough.

Results with Fully-Connected Classifiers: We use cosine classifiers in the main results. Table 7 shows detection results using fully-connected classifiers. Hallucination improves CoRPNs in all 1-shot cases, suggesting the *effectiveness of hallucination irrespective of the classifier choice*.

Performance Gain Analysis: We investigate where our performance gain comes from. The detection AP can be improved in two ways: (1) more boxes are correctly identified, *i.e.*, true positive rate goes up; (2) fewer boxes are misclassified, *i.e.*, false positive rate goes down. As shown

Method	Novel Set 1	Novel Set 2	Novel Set 3
CoRPNs w/ fc [48]	40.8	20.4	29.4
CoRPNs w/ fc + Halluc	**44.2**	**23.0**	**31.5**

Table 7. 1-shot detection performance (AP50) on PACAL VOC novel classes with fully-connected classifiers ('w/ fc'). CoRPNs and 'CoRPNs + Halluc' use the same set of hyper-parameters and share the same RPN outputs. 'CoRPNs + Halluc' outperforms CoRPNs under all three base/novel splits.

Novel Set 1	Avg. True Positive Boxes (↑)	Avg. False Positive Boxes (↓)
TFA [38]	118.9	4096.6
TFA + Halluc	**132.0**	**3309.3**
CoRPNs [48]	133.9	3440.6
CoRPNs + Halluc	**137.0**	**3041.1**
Novel Set 2		
TFA [38]	**76.8**	897.1
TFA + Halluc	69.5	**844.9**
CoRPNs [48]	**78.7**	920.1
CoRPNs + Halluc	77.1	**826.6**
Novel Set 3		
TFA [38]	109.3	2043.2
TFA + Halluc	**113.6**	**1461.4**
CoRPNs [48]	110.3	1704.7
CoRPNs + Halluc	**117.6**	**1345.0**

Table 8. Average total number of true positive boxes and false positive boxes in novel classes for different 1-shot models on PASCAL VOC. Hallucination increases the number of true positive boxes in most cases and decreases the number of false positive boxes, with a larger improvement in the false positive rate.

in Table 8, hallucination *both* increases the number of true positives in most cases and always decreases the number of false positives. We find that the scale of improvements on false positives is larger than the scale of improvements on true positives. A possible explanation is that hallucination generates *close-to-boundary* examples, and *improves decision boundaries*. Many misclassified boxes are eliminated, since we have a *better estimation of decision boundaries*.

5. Conclusion and Future Work

This paper tackles the lack of sample variation problem in few-shot detection. We introduce a hallucinator network to generate additional feature-level training examples. Experimental evaluation demonstrates that data hallucination, as a general strategy, can be incorporated into different types of detectors and substantially improves few-shot detection. Our analysis further shows that different design choices will change the behavior of the hallucinator.

Our method is one possible instantiation of the data hallucination strategy. We believe that further refinements of the proposed hallucinator architectures and exploration of a more dedicated control of training procedure could lead to a more effective hallucinator with better performance. Finally, it is a promising avenue to continue this line of work for other visual recognition tasks beyond object detection.

References

[1] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. YOLOv4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020. 2

[2] Hao Chen, Yali Wang, Guoyou Wang, and Yu Qiao. LSTD: A low-shot transfer detector for object detection. In *AAAI*, 2018. 1, 2

[3] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *ICCV*, 2017. 2

[4] Xuanyi Dong, Liang Zheng, Fan Ma, Yi Yang, and Deyu Meng. Few-example object detection with model communication. *IEEE TPAMI*, 41(7):1641–1654, 2019. 1

[5] Ian Endres and Derek Hoiem. Category independent object proposals. In *ECCV*, 2010. 2

[6] Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. The PASCAL visual object classes (VOC) challenge. *IJCV*, 88(2):303–338, 2010. 5

[7] Qi Fan, Wei Zhuo, Chi-Keung Tang, and Yu-Wing Tai. Few-shot object detection with attention-RPN and multi-relation detector. In *CVPR*, 2020. 2, 3

[8] Li Fei-Fei, Rob Fergus, and Pietro Perona. One-shot learning of object categories. *IEEE TPAMI*, 28(7):594–611, 2006. 1

[9] Michael Fink. *Acquiring a new class from a few examples: Learning recurrent domain structures in humans and machines*. PhD thesis, The Hebrew University of Jerusalem, 2011. 1

[10] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017. 1

[11] Hang Gao, Zheng Shou, Alireza Zareian, Hanwang Zhang, and Shih-Fu Chang. Low-shot learning via covariance-preserving adversarial augmentation networks. In *NeurIPS*, 2018. 3

[12] Ross Girshick. Fast R-CNN. In *ICCV*, 2015. 2

[13] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014. 1, 2

[14] Bharath Hariharan and Ross Girshick. Low-shot visual recognition by shrinking and hallucinating features. In *ICCV*, 2017. 3

[15] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *ICCV*, 2017. 2

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE TPAMI*, 37(9):1904–1916, 2015. 2

[17] Ting-I Hsieh, Yi-Chen Lo, Hwann-Tzong Chen, and Tyng-Luh Liu. One-shot object detection with co-attention and co-excitation. In *NeurIPS*, 2019. 3, 5, 6

[18] Bingyi Kang, Zhuang Liu, Xin Wang, Fisher Yu, Jiashi Feng, and Trevor Darrell. Few-shot object detection via feature reweighting. In *ICCV*, 2019. 1, 3, 5

[19] Hei Law and Jia Deng. CornerNet: Detecting objects as paired keypoints. In *ECCV*, 2018. 2

[20] Bohao Li, Boyu Yang, Chang Liu, Feng Liu, Rongrong Ji, and Qixiang Ye. Beyond max-margin: Class margin equilibrium for few-shot object detection. In *CVPR*, 2021. 2, 6

[21] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017. 2, 3, 5

[22] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common objects in context. In *ECCV*, 2014. 5

[23] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: Single shot multibox detector. In *ECCV*, 2016. 2

[24] Anton Osokin, Denis Sumin, and Vasily Lomakin. OS2D: One-stage one-shot object detection by matching anchor features. In *ECCV*, 2020. 3

[25] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *CVPR*, 2016. 2

[26] Joseph Redmon and Ali Farhadi. YOLO9000: Better, Faster, Stronger. In *CVPR*, 2017. 2

[27] Joseph Redmon and Ali Farhadi. YOLOv3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018. 2

[28] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NeurIPS*, 2015. 1, 2, 5

[29] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet large scale visual recognition challenge. *IJCV*, 115(3):211–252, 2015. 1, 3

[30] Ruslan Salakhutdinov, Joshua Tenenbaum, and Antonio Torralba. One-shot learning with a hierarchical nonparametric Bayesian model. In *ICML Workshop on Unsupervised and Transfer Learning*, 2012. 2

[31] Lauren A Schmidt. *Meaning and compositionality as statistical induction of categories and constraints*. PhD thesis, Massachusetts Institute of Technology, 2009. 1

[32] Eli Schwartz, Leonid Karlinsky, Joseph Shtok, Sivan Harary, Mattias Marder, Abhishek Kumar, Rogerio Feris, Raja Giryes, and Alex Bronstein. Delta-encoder: An effective sample synthesis method for few-shot object recognition. In *NeurIPS*, 2018. 3

[33] Eli Schwartz, Leonid Karlinsky, Joseph Shtok, Sivan Harary, Mattias Marder, Sharathchandra Pankanti, Rogerio Feris, Abhishek Kumar, Raja Giries, and Alex M. Bronstein. RepMet: Representative-based metric learning for classification and one-shot object detection. In *CVPR*, 2019. 1, 2

[34] Jake Snell, Kevin Swersky, and Richard S Zemel. Prototypical networks for few-shot learning. In *NeurIPS*, 2017. 1, 8

[35] Bo Sun, Banghuai Li, Shengcai Cai, Ye Yuan, and Chi Zhang. FSCE: Few-shot object detection via contrastive proposal encoding. In *CVPR*, 2021. 3, 6

[36] Koen van de Sande, Jasper Uijlings, Theo Gevers, and Arnold Smeulders. Segmentation as selective search for object recognition. In *ICCV*, 2011. 2

[37] Oriol Vinyals, Charles Blundell, Timothy P. Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. In *NeurIPS*, 2016. 1

[38] Xin Wang, Thomas E. Huang, Trevor Darrell, Joseph E Gonzalez, and Fisher Yu. Frustratingly simple few-shot object detection. In *ICML*, 2020. 1, 2, 3, 5, 6, 7, 8

[39] Yu-Xiong Wang, Deva Ramanan, and Martial Hebert. Meta-learning to detect rare objects. In *ICCV*, 2019. 1, 2, 3

[40] Yu-Xiong Wang, Ross Girshick, Martial Hebert, and Bharath Hariharan. Low-shot learning from imaginary data. In *CVPR*, 2018. 2, 3, 4, 6

[41] Yu-Xiong Wang and Martial Hebert. Model recommendation: Generating object detectors from few samples. In *CVPR*, 2015. 1

[42] Yu-Xiong Wang and Martial Hebert. Learning to learn: Model regression networks for easy small sample learning. In *ECCV*, 2016. 1

[43] Jiaxi Wu, Songtao Liu, Di Huang, and Yunhong Wang. Multi-scale positive sample refinement for few-shot object detection. In *ECCV*, 2020. 2, 3, 6, 7

[44] Yang Xiao and Renaud Marlet. Few-shot object detection and viewpoint estimation for objects in the wild. In *ECCV*, 2020. 2, 3, 6, 7

[45] Xiaopeng Yan, Ziliang Chen, Anni Xu, Xiaoxi Wang, Xiaodan Liang, and Liang Lin. Meta R-CNN: Towards general solver for instance-level low-shot learning. In *ICCV*, 2019. 2, 3, 6

[46] Yukuan Yang, Fangyun Wei, Miaojing Shi, and Guoqi Li. Restoring negative information in few-shot object detection. In *NeurIPS*, 2020. 2, 6

[47] Ruixiang Zhang, Tong Che, Zoubin Ghahramani, Yoshua Bengio Bengio, and Yangqiu Song. MetaGAN: An adversarial approach to few-shot learning. In *NeurIPS*, 2018. 3

[48] Weilin Zhang, Yu-Xiong Wang, and David A. Forsyth. Cooperating RPN's improve few-shot object detection. *arXiv preprint arXiv:2011.10142*, 2020. 2, 3, 4, 5, 6, 7, 8

[49] Xingyi Zhou, Jiacheng Zhuo, and Philipp Krahenbuhl. Bottom-up object detection by grouping extreme and center points. In *CVPR*, 2019. 2

[50] Chenchen Zhu, Fangyi Chen, Uzair Ahmed, and Marios Savvides. Semantic relation reasoning for shot-stable few-shot object detection. In *CVPR*, 2021. 3, 6

[51] Xiangxin Zhu, Dragomir Anguelov, and Deva Ramanan. Capturing long-tail distributions of object subcategories. In *CVPR*, 2014. 1