NIFF: Alleviating Forgetting in Generalized Few-Shot Object Detection via Neural Instance Feature Forging

Karim Guirguis[1,2†] Johannes Meier[1†] George Eskandar[3] Matthias Kayser[1] Bin Yang[3] Jürgen Beyerer[2,4]
Robert Bosch GmbH[1] Karlsruhe Institute of Technology[2] University of Stuttgart[3] Fraunhofer IOSB[4]

Abstract

Privacy and memory are two recurring themes in a broad conversation about the societal impact of AI. These concerns arise from the need for huge amounts of data to train deep neural networks. A promise of Generalized Few-shot Object Detection (G-FSOD), a learning paradigm in AI, is to alleviate the need for collecting abundant training samples of novel classes we wish to detect by leveraging prior knowledge from old classes (i.e., base classes). G-FSOD strives to learn these novel classes while alleviating catastrophic forgetting of the base classes. However, existing approaches assume that the base images are accessible, an assumption that does not hold when sharing and storing data is problematic. In this work, we propose the first data-free knowledge distillation (DFKD) approach for G-FSOD that leverages the statistics of the region of interest (RoI) features from the base model to forge instance-level features without accessing the base images. Our contribution is three-fold: (1) we design a standalone lightweight generator with (2) class-wise heads (3) to generate and replay diverse instance-level base features to the RoI head while finetuning on the novel data. This stands in contrast to standard DFKD approaches in image classification, which invert the entire network to generate base images. Moreover, we make careful design choices in the novel finetuning pipeline to regularize the model. We show that our approach can dramatically reduce the base memory requirements, all while setting a new standard for G-FSOD on the challenging MS-COCO and PASCAL-VOC benchmarks.

1. Introduction

Object detection (OD) is an integral element in modern computer vision perception systems (e.g., robotics and self-driving cars). However, object detectors [1–8] require abundant annotated data to train, which is labor and time intensive. In some applications requiring rare class detection, collecting much data is challenging. Striving to learn in limited data scenarios, few-shot object detection (FSOD) [9] is an uprising field. It mimics the human cognitive ability by leveraging prior knowledge from previous experiences with abundant base data, to rapidly learn novel classes from limited samples. Despite the success of meta-learning [10–15] and transfer learning [16–19] paradigms in FSOD, most methods prioritize the detection performance of the novel classes while ignoring the catastrophic forgetting of the base ones. This might lead to critical failure cases in real-life operational perception systems.

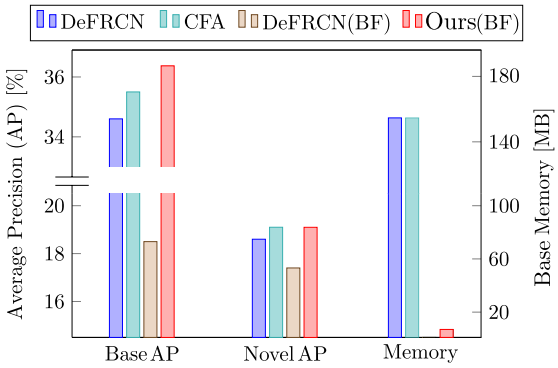

Figure 1. The base memory requirements for G-FSOD are dramatically reduced by our framework, while improving the overall detection performance on MS-COCO (10-shot). We only store a lightweight generator that synthesizes deep features for the RoI head. BF denotes base-data free finetuning.

To address the aforementioned concern, generalized few-shot object detection (G-FSOD) has been introduced to jointly detect the base and novel classes. One of the first approaches to address the G-FSOD task was TFA [16], which finetunes the detector using a balanced set of base and novel class samples while freezing the backbone and the region proposal network (RPN). While this has reduced forgetting, the performance on novel classes has dropped significantly. Since then, a plethora of works have attempted to improve the overall detection performance. DeFRCN [19] proposed a gradient manipulation approach to modify the RPN and RoI head gradients. Retentive R-CNN [22], a knowledge distillation approach and CFA [23], a gradient manipulation approach were proposed to explicitly tackle the catastrophic forgetting of base classes. However, all the above approaches rely on the assumption that base data is available while learning the new classes. This made us raise the following question: *How to alleviate forgetting without base data in case of a memory constraint or privacy concerns restricting the storage and replay of old base data?*

† Authors have equally contributed to this work.
Corresponding author: karim.guirguis@de.bosch.com

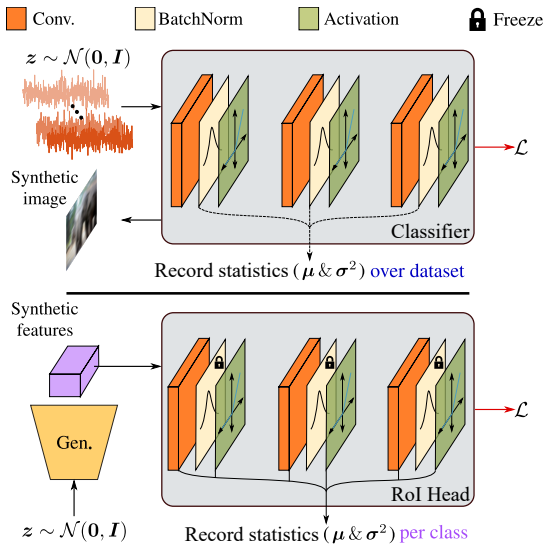

Figure 2. **Top:** Standard DFKD approach via inverting the entire model [20, 21]. **Bottom:** An overview of our proposed approach. We abstractly show a few layers in both models. The main differences are: (1) we synthesize features instead of images, (2) we use a separate generator instead of inverting the model, and (3) we record the class-wise statistics (instead of the full data statistics) before and after the normalization layers in the RoI head.

Data-free knowledge distillation (DFKD) is a line of work sharing a similar interest in transferring knowledge without storing raw data. DeepDream [24] pioneered model inversion (MI) work which inverted a pre-trained classifier to generate synthetic images for knowledge distillation. Since then, various works [20, 21, 25] have followed attempting to generate higher-fidelity images and even in a class-incremental setting [21]. Despite its success in image classification, applying DFKD in G-FSOD comes with several challenges. First, most works revolve around generating synthetic images via MI. In the context of OD and G-FSOD, this entails generating images with bounding boxes which inflicts higher computational and memory overhead. Although a recent approach DIODE [25] has applied MI in OD, it cannot be extended to G-FSOD for the following reason. Similar to all the previously mentioned works in DFKD, DIODE needs the statistics of the BatchNorm(BN) [26] layers which are trained on the detection datasets. However, the backbone in G-FSOD is pre-trained on ImageNet and frozen (except the last ResBlock) during the entire training (unfreezing would change the mature parameters and will reduce the overall performance). Hence, the running means and variances in the BN do not represent the true base data distribution.

Contribution: In this work, we propose Neural Instance Feature Forging (NIFF), the first data-free knowledge distillation method for G-FSOD. We aim to alleviate forgetting without storing base data to respect privacy restrictions and reduce the overall memory footprint, as shown in Fig. 1. Our two key insights are as follows. First, we show that the statistics of instance-level RoI head features sufficiently represent the distribution of base classes. Second, we show that a standalone lightweight generator can be trained in a distillation fashion to match the gathered statistics and synthesize class-wise base features to train a G-FSOD model. This stands in contrast to MI approaches which optimize the pre-trained model to synthesize high-fidelity images. Our contributions are summarized as follows:

1. We forge instance-level features instead of synthesizing images (Fig. 2) as the feature space ($1024 \times 7 \times 7$) is much smaller than the image space ($3 \times 600 \times 1000$).
2. Rather than inverting the whole model, instead we design a standalone lightweight generator in the feature space to forge instance-level base features. The generator is able to better capture the feature distribution than the complex MI.
3. We equip the generator with class-aware heads rather than a class-agnostic one, and we train it to synthesize features with a distribution that matches the class-wise statistics of the pre-trained base RoI head, hence promoting feature diversity. We demonstrate that class-aware heads trained on class-aware statistics outperform a shared model trained on class-wise statistics.
4. We dramatically reduce the overall memory footprint by two orders of magnitude while setting a new standard for the overall detection performance in G-FSOD on MS-COCO [27] and PASCAL-VOC [28]. For instance, the base images and annotations for MS-COCO dataset (10-shot) are $\sim$ 150MB large, while our generator parameters only occupy $\sim$ 4MB.

2. Related Works

Few-Shot Object Detection. FSOD strategies can be divided into meta-learning and transfer learning approaches. The meta-learning methods gain knowledge by solving a variety of unrelated tasks [10–15]. On the other hand, the transfer learning approaches directly finetune on the novel data. On a balanced training set of both base and novel classes, TFA [16] finetunes just the box predictor. By creating multi-scale positive samples as object pyramids, MPSR [17] addresses the high scale variances by improving the prediction at many scales. DeFRCN [19] highlights the contradictory goals of the RPN and class-aware ROI head. Accordingly, they remove the RPN gradients and downscale the ROI head gradients flowing to the backbone. However, the main goal of FSOD approaches is to boost the performance on the novel classes.

Generalized Few-Shot Object Detection. A growing area of study within FSOD is G-FSOD, which focuses on detecting both the base and novel classes. TFA [16] attempts to prevent forgetting by optimizing a balanced set of base and novel classes, whereas ONCE [29] tackles the

issue in an incremental class learning setting using a meta-learning strategy. A transfer-learning-based strategy, Retentive R-CNN [22], leverages the base-trained model in a distillation-like fashion to alleviate forgetting. To reduce the extra computational and memory costs, CFA [23] developed a new gradient update mechanism to alleviate forgetting based on the angle between gradients for base and novel samples. All of the approaches outlined above, however, are heavily reliant on the availability of annotated base images during the novel training phase.

Data-Free Knowledge Distillation. DeepDream [24] was the first among this line of work to show that discriminative neural networks harbor information that enables the generation of images. It synthesizes an image to provide a high output response for specific classes at different model layers. Later, DeepInversion [20] improved the dreamed image quality by penalizing the distance between statistics of the features, assuming a Gaussian distribution. AlwaysBeDreaming (ABD) [21] minimizes the feature drift over the previous task while finetuning the last classification layer with a cross entropy loss. Recently, Chawla et al. [25] proposed a non class incremental data-free knowledge distillation approach for YOLOv3 based on MI along with a bounding box sampling scheme. If extended to G-FSOD, the backbone and BN layers would be unfrozen, which would limit the model's capacity to rapidly learn new classes. Moreover, BN layers do not capture the true class-wise distribution.

3. Methodology

We aim to design a G-FSOD pipeline that learns novel classes from scant data while preserving privacy and memory constraints. In this section, we start by formally defining the G-FSOD problem. Then, we revisit the data-free knowledge distillation via noise optimization. Finally, we present our approach, NIFF, which consists of two stages: (1) Feature generator training and (2) Novel training in a distillation fashion via the trained generator.

3.1. Problem Formulation

Analogous to FSOD, G-FSOD divides the dataset $\mathcal{D}$ into a base dataset $\mathcal{D}_b$ and a novel dataset $\mathcal{D}_n$, with abundant instances of base classes $\mathcal{C}_b$ and a limited number of instances of novel classes $\mathcal{C}_n$ (i.e., $\mathcal{C}_b \cap \mathcal{C}_n = \emptyset$), respectively. Each input image $x \in \mathcal{X}$ is associated with an annotation $y \in \mathcal{Y}$ comprising the class label c_i and the corresponding bounding box coordinates b_i for each instance i. Formally, $\mathcal{D}_b = \{ (x, y) \mid y = \{(c_i, b_i)\}, c_i \in \mathcal{C}_b\}$, and $\mathcal{D}_n = \{ (x, y) \mid y = \{(c_i, b_i)\}, c_i \in \mathcal{C}_n\}$.

The training of a G-FSOD consists of two main stages. Firstly, the base training stage strives to build a strong knowledge prior by training on $\mathcal{D}_b$. Secondly, novel training leverages the acquired knowledge to learn the novel classes from $\mathcal{D}_n$ rapidly. Unlike FSOD, G-FSOD aims to maintain its ability to detect $\mathcal{C}_b$ while learning $\mathcal{C}_n$ by leveraging base data samples. G-FSOD performance is measured by the overall Average Precision (AP), which is calculated as a weighted average of the base classes AP (bAP) and novel classes AP (nAP):

$$AP = \frac{|\mathcal{C}_b| \cdot bAP + |\mathcal{C}_n| \cdot nAP}{|\mathcal{C}_b| + |\mathcal{C}_n|} \tag{1}$$

The goal of G-FSOD is to maximize the AP. However, we raise the following arguments: (1) Since the base data samples utilized during novel training are fixed K-shots, they do not represent the full base distribution, (2) It is not always possible to store base data samples due to privacy and/or memory constraints. While recent attempts have been made to tackle the former problem [22, 23], the latter has not been investigated.

3.2. Revisiting Data-Free Knowledge Distillation

DFKD approaches aim to distill and transfer knowledge from a teacher to a student network by synthesizing images as an alternative to the old tasks' data. The most common approaches reviewed in Sec. 2 are based on a two-step noise optimization paradigm. First, a noise vector is sampled from a Gaussian distribution and iteratively optimized into a synthesized image with stochastic gradient descent (SGD). This is realized by minimizing the Kullback–Leibler (KL) divergence between the gathered statistics and the statistics yielded by the synthetic images under a Gaussian assumption. The second stage employs standard data-driven knowledge distillation approaches using the synthetic images from the first stage. The aim is to transfer knowledge in a teacher-student fashion to alleviate forgetting.

To this point, we highlight two main challenges. Firstly, how can we accomplish DFKD in G-FSOD using BN statistics while the majority of the backbone and all BN layers are frozen? Unfreezing such layers during finetuning on the base and novel datasets would lead to parameters and batch statistics that are presumably different from the pre-trained ImageNet ones. As a result, the network will learn new parameters and statistics that differ from the more mature pre-trained ones, devaluing the overall detection performance. Moreover, the BN statistics do not depict the true class-wise means and variances. Secondly, different from image classification, multiple instances per image and the RPN make it challenging to invert the model prior to the RoI head. Otherwise, generating bounding boxes would be needed for the synthesized images incurring higher complexity and significantly more computational and memory overhead.

3.3. Stage I: Feature Generator Training

Our NIFF framework generates instance-level base features to be replayed during novel class learning. In the first

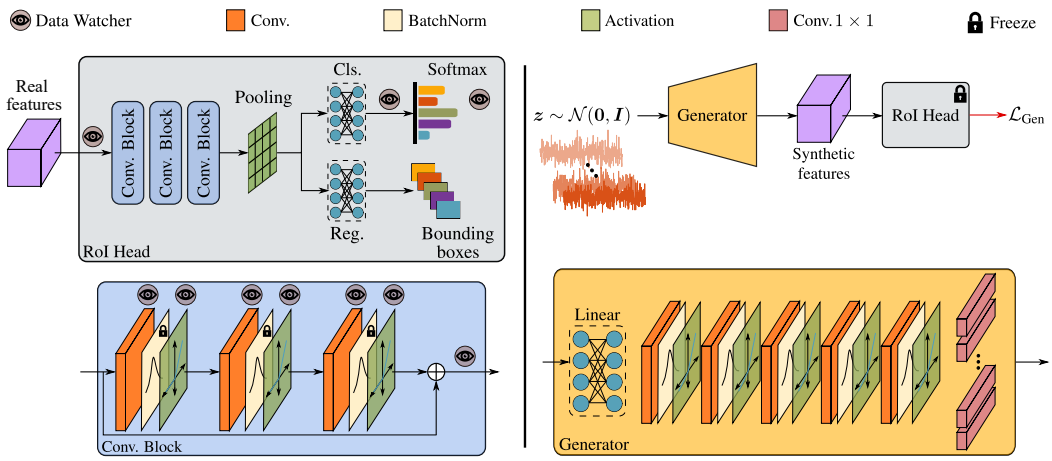

Figure 3. **Stage I: Feature Generator Training. Left:** A detailed overview of the RoI head to highlight where the features' statistics are gathered using the data watchers. **Right:** Illustration of the proposed generator training pipeline and the architectural details.

stage, we train a standalone feature generator by aligning the class-wise means and variances at the RoI head. In the second stage, we synthesize features from the generator during novel training to alleviate forgetting and make careful considerations in the training of the pipeline to regularize the detector. In this section, we describe the first stage of our approach addressing the generator design and training.

Gathering base statistics. Since the BN layers are frozen in FSOD and G-FSOD models, an alternative way is needed to gather meaningful statistics (running means and variances) of the base RoI features. Differently from the discussed DFKD methods, we choose to record the class-wise statistics as opposed to class agnostic statistics. This design choice allows more fine-grained control over the number and type of the generated features. A class conditional generation can compensate for the sparser and harder classes in the base dataset. To this end, we introduce the *data watcher* block which performs average pooling on the spatial dimensions of the input feature maps and records the class-wise mean $\boldsymbol{\mu_c}$ and variance $\boldsymbol{\sigma_c^2}$ vectors and the sample size n_c. Formally, we update the statistics via combined mean and corrected variance as follows:

$$\hat{\boldsymbol{\mu}}_{\boldsymbol{c,t}} = \frac{\hat{n}_{c,t-1}\hat{\boldsymbol{\mu}}_{\boldsymbol{c,t-1}} + n_{c,t}\boldsymbol{\mu}_{\boldsymbol{c,t}}}{\hat{n}_{c,t-1} + n_{c,t}}, \tag{2}$$

$$\begin{aligned}\hat{\boldsymbol{\sigma}}^2_{\boldsymbol{c,t}} = &\frac{(\hat{n}_{c,t-1}-1)\hat{\boldsymbol{\sigma}}^2_{\boldsymbol{c,t-1}} + (n_{c,t}-1)\boldsymbol{\sigma}^2_{\boldsymbol{c,t}}}{\hat{n}_{c,t-1} + n_{c,t} - 1} \\ &+ \frac{\hat{n}_{c,t-1}n_{c,t}(\boldsymbol{\mu}_{\boldsymbol{c,t}} - \hat{\boldsymbol{\mu}}_{\boldsymbol{c,t-1}})}{(\hat{n}_{c,t-1} + n_{c,t})(\hat{n}_{c,t-1} + n_{c,t} - 1)},\end{aligned} \tag{3}$$

where t denotes the iteration step. $\hat{(.)}$ denotes a running estimate. To restrict the generator to forge more diverse features, we opt to place the data watchers at various layers in the RoI head (i.e., before the frozen BN layers and after the activations), as shown in Fig. 3 (left). The more data watchers we place, the more restricted the forged features are. For this, we further place data watchers before and after the softmax. The collected statistics should be able to depict a strong prior distribution of the base RoI features. It is essential to highlight that once the statistics are collected for the base task, the data is no longer seen or stored. This means that the model enclosing the underlying task statistics can be treated as a black box maintaining data privacy, allowing to share the model with different parties without sharing or storing the data. Note that relying solely on the running averages of the RoI stats results in losing information, especially since we only use RoI pooled features (1024x7x7 fixed output), making it hard to reconstruct the training data. NIFF requires even fewer stats than previous methods like DIODE [25] (which uses backbone stats) and hence cannot reconstruct an entire image.

Generator architecture. Next, we leverage the gathered means and variances of the RoI features to train the proposed feature generator with SGD. As shown in Fig. 3 (right), our proposed light-weight generator architecture consists of a linear layer to map the input noise vector $\boldsymbol{z} \in \mathbb{R}^{100}$ into $\mathbb{R}^{392}$ which are then reshaped to $\mathbb{R}^{8\times7\times7}$ and fed to 5 sequential convolutional blocks. Each block comprises of Conv2D layers with 8 channels and kernel size 3. Finally, we append a number of $|\mathcal{C}_b|$ 1×1 convolutional blocks in parallel to enable generating class-wise instance features $\boldsymbol{f_c} \in \mathbb{R}^{1024\times7\times7}$. After sampling noise $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ we generate synthetic features for class c via $\boldsymbol{f_c} = G(c, \boldsymbol{z})$, where G is our generator model.

Generator training by aligning class-wise statistics. The generator is trained to forge instance-level base features by aligning the acquired statistics in the RoI head at the same layers. The statistics alignment forces the generator to produce diverse base features. To force the generator to produce class-specific features $\boldsymbol{f_c}$ from the different heads, we align the class-wise statistics obtained by passing $\boldsymbol{f_c}$ to the RoI head with the class-wise statistics gathered

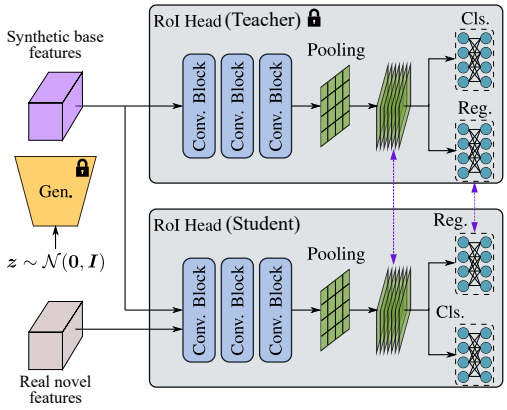

Figure 4. **Stage II: Improved Novel Training Pipeline.** During novel training, we perform knowledge distillation via the the trained base feature generator at the RoI head.

by the data watchers. This is opposed to aligning the class-agnostic full dataset statistics. To further steer each separate head to produce different f_c, we add a cross-entropy loss between the corresponding target class-label $\boldsymbol{y_{i,c}}$ and the probability $\boldsymbol{p_{i,c}}$ at the final softmax layer.

In summary, the generator is trained using two main objectives: (1) Align the RoI head statistics with the gathered base ones via KL divergence under a Gaussian assumption, and (2) Maximize the class probability via cross-entropy loss. The generator loss function $\mathcal{L}_{\text{Gen}}$ is denoted by:

$$\mathcal{L}_{\text{Gen}} = \lambda_{KL} \frac{1}{|\mathcal{C}_b| * d} \sum_{c=1}^{|\mathcal{C}_b|} \sum_{i=1}^{d} \log \frac{\tilde{\boldsymbol{\sigma}}_{\boldsymbol{c,i}}}{\boldsymbol{\sigma}_{\boldsymbol{c,i}}} - \frac{1}{2}\left[1 - \frac{\boldsymbol{\sigma}_{\boldsymbol{c,i}}^2 + (\tilde{\boldsymbol{\mu}}_{\boldsymbol{c,i}} - \boldsymbol{\mu}_{\boldsymbol{c,i}})^2}{\tilde{\boldsymbol{\sigma}}_{\boldsymbol{c,i}}^2}\right] - \frac{1}{|\mathcal{C}_b| * N} \sum_{i=1}^{|\mathcal{C}_b|*N} \frac{1}{|\mathcal{C}_b|} \sum_{c=1}^{|\mathcal{C}_b|} \boldsymbol{y_{i,c}} \log(\boldsymbol{p_{i,c}}). \quad (4)$$

where λ_{KL} is a weighting factor. This loss is averaged over all Data Watchers (omitted for better readability). $\boldsymbol{y_i}$ is a one-hot ground-truth vector. d is the pooled feature dimension. $\boldsymbol{\mu_c}$ and $\boldsymbol{\sigma_c^2}$ are the gathered feature statistics, while $\tilde{\boldsymbol{\mu}}_{\boldsymbol{c}}$ and $\tilde{\boldsymbol{\sigma}}_{\boldsymbol{c}}^{\boldsymbol{2}}$ are the fake features statistics. N is the total number of generated features per class. During training, we set $N = 600$. To avoid memory overflow during training, we feed N features for each class sequentially, accumulate the gradients and backpropagate once at the end.

3.4. Stage II: Improved Novel Training Pipeline

The final step of NIFF is to finetune on novel data while performing knowledge distillation at the RoI head in a teacher-student fashion. As depicted in Fig. 4, along with the real novel features we additionally feed forged base instance-level features via the trained generator network. To match the finetuning K-shot setting, we set $N = K$ features per class, which means that all base classes are encountered throughout each iteration. This presents an important advantage in comparison to the data-dependent approaches like [22, 23] which finetune with base images that contain a few classes only in each iteration. Another advantage is that our approach is generative: this means we can sample from z to produce diverse features, whereas in [22, 23] a fixed number of shots for each base class is presented during training limiting the seen distribution.

The distillation is performed as follows: First, we employ a weighted feature distillation using L2-norm to penalize the difference between class-wise pooled RoI features of the teacher $\boldsymbol{F^T} \in \mathbb{R}^{(|\mathcal{C}_b|*K)\times d}$ and student $\boldsymbol{F^S} \in \mathbb{R}^{(|\mathcal{C}_b|*K)\times d}$, where d is the pooled feature dimension. Since we generate class-wise features, we weight the difference between the features with the weight vector $\boldsymbol{W}_{\text{Cls}}^{\boldsymbol{c}} \in \mathbb{R}^d$ of the entire classification weight matrix $\boldsymbol{W}_{\text{Cls}} \in \mathbb{R}^{|\mathcal{C}_b|\times d}$ for the corresponding class c. We also weight the regression feature differences in the same way with the corresponding weight matrix $\boldsymbol{W}_{\text{Reg}}^{\boldsymbol{c}} \in \mathbb{R}^{4\times d}$ from the regression weight tensor $\boldsymbol{W}_{\text{Reg}} \in \mathbb{R}^{(4*|\mathcal{C}_b|)\times d}$. Secondly, we align the class-wise regression logits by penalizing the drift between the predicted offsets of the teacher $\boldsymbol{reg_c^T} \in \mathbb{R}^4$ and student $\boldsymbol{reg_c^S} \in \mathbb{R}^4$. The distillation loss can be written as:

$$\begin{aligned}\mathcal{L}_{\text{KD}} = \lambda_F \frac{1}{|\mathcal{C}_b| * K} \sum_{i=1}^{|\mathcal{C}_b|*K} \|(\boldsymbol{F_i^T} - \boldsymbol{F_i^S}) * \boldsymbol{W}_{\text{Cls}}^{\boldsymbol{c_i}}\|_2^2 \\ + \lambda_F \frac{1}{4 * |\mathcal{C}_b| * K} \sum_{i=1}^{|\mathcal{C}_b|*K} \sum_{j=1}^{4} \|(\boldsymbol{F_i^T} - \boldsymbol{F_i^S}) * \boldsymbol{W}_{\text{Reg}}^{\boldsymbol{c_i,j}}\|_2^2 \\ + \frac{1}{|\mathcal{C}_b| * K} \sum_{i=1}^{|\mathcal{C}_b|*K} \|\boldsymbol{reg_i^T} - \boldsymbol{reg_i^S}\|_1,\end{aligned} \quad (5)$$

where λ_F is a weighting factor for the feature distillation. Thirdly, similar to [21], we employ a cross-entropy loss during finetuning to maximize the forged features confidence:

$$\mathcal{L}_{\text{conf}} = -\frac{1}{|\mathcal{C}_b| * K} \sum_{i=1}^{|\mathcal{C}_b|*K} \frac{1}{|\mathcal{C}|} \sum_{c=1}^{|\mathcal{C}|} \boldsymbol{y_{i,c}} \log(\boldsymbol{p_{i,c}}). \quad (6)$$

The overall novel training loss $\mathcal{L}_{\text{N}}$ can be denoted by:

$$\mathcal{L}_{\text{N}} = \mathcal{L}_{\text{Cls}} + \mathcal{L}_{\text{Reg}} + \mathcal{L}_{\text{KD}} + \mathcal{L}_{\text{conf}}, \quad (7)$$

where $\mathcal{L}_{\text{Cls}}$ and $\mathcal{L}_{\text{Reg}}$ denote the cross-entropy and smooth L1 losses, respectively [3].

Additional regularization. To this point, we find that by replaying the generated base features, the proposed model is almost on par with the state-of-the-art in the overall AP. During novel training, we find that there are important design choices in the training pipeline which can boost the overall detection performance. We opt to perform the recently proposed constraint finetuning approach (CFA) [23]. With the forged base features available, we are able to leverage the backpropagated base gradients to apply CFA. Additionally, we investigate various pixel-level and parameter-level regularization techniques. For the former, we utilize random color jittering (i.e., brightness, contrast, and saturation), random flipping, and random cropping. All augmentations are applied with a probability $p_{\text{aug}} = 0.5$.

For parameter-level regularization, we find that employing the elastic weight consolidation (EWC) [30] approach, which was originally designed for image classification, can alleviate forgetting. EWC weights the importance of the parameters by computing the diagonal of the Fisher information matrix (FIM). We compute the FIM by squaring the backpropagated gradients during the last epoch in the base training. During the novel training, EWC penalizes the change of important parameters dictated by FIM to allow for a more effective knowledge transfer. However, the FIM occupies a large memory space as it saves a weight for each model parameter. To mitigate this undesirable effect, we average each layer's weights in the FIM, reducing the memory from $\sim$ 200MB to 6.8KB. Thus, each layer's importance is represented by a single scalar value. We denote this approximation of EWC by mean EWC (mEWC). We show in the supplementary that mEWC is on par with EWC when applied to our model.

4. Experiments

We evaluate our proposed approach using the well-established G-FSOD benchmarks, including MS-COCO [27] and PASCAL-VOC [28] datasets. We employ the same data splits as in earlier works [16, 19, 23] to ensure a fair comparison. Implementation details are provided in the Supplementary material.

Datasets. Firstly, the MS-COCO dataset contains 80 classes, of which 60 are base categories disjoint with PASCAL-VOC, and the remaining 20 are unique classes. We use $5k$ validation set during testing, while the rest is used for training. We report the outcomes of $K = 5, 10, 30-$shot settings. Secondly, the PASCAL-VOC dataset has three distinct splits, each with 20 classes. Further, the classes are randomly divided into 15 and 5, respectively, base and novel classes. For base and novel training, data is drawn from the VOC 2007 and VOC 2012 train/val sets. The VOC 2007 test set is used for testing. The outcomes of $K = 1, 2, 3, 5, 10$-shot settings are reported.

Evaluation metrics. Following prior G-FSOD works [22, 23], we report the overall (AP), base (bAP), and novel (nAP) metrics. We also compute the Average Recall (AR) for the base (bAR) and novel (nAR) classes. Finally, we report the ensemble-inference results (w/E), leveraging the base model parameters during inference [22, 23].

4.1. Ablation Studies

In order to showcase the importance of our novel contributions and design choices, we ablate different parts of the pipeline on the MS-COCO dataset under the 10-shot setting.

Generator design choices. In Tab. 1, we validate our generator design choices without any regularization (namely: *standalone generator*, *class-wise statistics*, *separate heads* and *number of channels per layer*). First, we invert the RoI head to produce instance-level features by minimizing a KL-loss with respect to the gathered base data statistics. Note that this can be considered as a straightforward extension of the standard MI approach ABD [21] to G-FSOD, albeit this model produces features not images. Next, we train a standalone generator with a shared head for all classes while minimizing the full base-data statistics. Although the generator adds a negligible memory overhead, it outperforms the inverted model, which supports our claim that a separate generator is easier to optimize. However, when trained with class-wise statistics the performance slightly drops. We then replace the shared head with separate class-aware heads and and minimize the full base statistics. We notice that this setting is on par with the generator w/ shared head in row 2. Only when we combine the separate heads with the class-wise statistics can we achieve the best overall performance, as the model can now better consider the inter-class variance. Solely extending the model by class-wise statistics or by class-wise heads reduces the overall performance. Afterwards, we experiment with the complete version of the generator but we remove the cross-entropy loss from Eq. (4) and observe a slight drop in the overall AP. In the lower part of the table, we study the trade-off between AP and memory by changing the number of channels per generator layer. Interestingly, we find that the model with 64 channels and the model with 8 channels perform similarly, so we choose the minimalist design to reduce the memory footprint.

Model Configuration	10-Shot Inference				
	AP	bAP	nAP	AR	Memory [MB]
Inverted RoI head	28.9	32.6	17.7	27.1	**0**
Gen. w/ shared head w/o classwise stats	29.3	33.1	17.8	27.3	1.6
Gen. w/ shared head w/ classwise stats	28.5	32.1	17.7	26.4	1.6
Gen. w/ separate heads w/o classwise stats	29.0	32.8	17.5	27.4	3.7
Gen. w/ separate heads w/ classwise stats **(Ours)**	**30.7**	**35.0**	17.8	28.6	3.7
Ours w/o cross-entropy term	30.0	34.1	17.6	28.6	3.7
Ours w/ (dim = 64)	**30.7**	**35.0**	17.8	**28.8**	28.7
Ours w/ (dim = 32)	30.5	34.8	**17.9**	28.6	14.0
Ours w/ (dim = 16)	**30.7**	34.9	**17.9**	28.7	7.1
Ours w/ (dim = 8)	**30.7**	**35.0**	17.8	28.6	3.7

Table 1. Impact of various generator design choices on MS-COCO (10-shot) without the added regularization. By memory, we mean the overhead storage needed other than the detection model.

Impact of generated features sampling. In Tab. 3, we study the impact of sampling techniques when generating features during novel finetuning (no regularization is performed). We start by generating only 1 feature per class, then we experiment with a higher number of features to verify its effect on the overall AP. We observe that once we reach 10 features per class which matches the novel fine-

Methods / Shots	w/E	w/B	5 shot			10 shot			30 shot		
			AP	bAP	nAP	AP	bAP	nAP	AP	bAP	nAP
FRCN-ft-full [16]	✗	✓	18.0	22.0	6.0	18.1	21.0	9.2	18.6	20.6	12.5
TFA w/ fc [16]	✗	✓	27.5	33.9	8.4	27.9	33.9	10.0	29.7	35.1	13.4
TFA w/ cos [16]	✗	✓	28.1	34.7	8.3	28.7	35.0	10.0	30.3	35.8	13.7
MPSR [17]	✗	✓	-	-	-	15.3	17.1	9.7	17.1	18.1	14.1
DeFRCN [19]	✗	✓	28.7	33.1	15.3	30.6	34.6	**18.6**	31.6	34.7	**22.5**
ONCE [29]	✗	✓	13.7	17.9	1.0	13.7	17.9	1.2	-	-	-
Meta R-CNN [10]	✗	✓	3.6	3.5	3.8	5.4	5.2	6.1	7.8	7.1	9.9
FSRW [31]	✗	✓	-	-	-	-	-	5.6	-	-	9.1
FsDetView [12]	✗	✓	5.9	5.7	6.6	6.7	6.4	7.6	10.0	9.3	12.0
CFA w/ fc [23]	✗	✓	**30.1**	**37.1**	9.0	30.8	**37.6**	10.5	31.9	**37.7**	14.7
CFA w/ cos [23]	✗	✓	29.7	**36.3**	9.8	30.3	**36.6**	11.3	31.7	37.0	15.6
CFA-DeFRCN [23]	✗	✓	**30.1**	35.0	**15.6**	**31.4**	35.5	**19.1**	**32.0**	35.0	**23.0**
DeFRCN	✗	✗	23.7	26.3	**15.6**	18.2	18.5	17.4	16.3	16.3	21.4
NIFF-DeFRCN	✗	✗	**31.3**	**36.3**	**15.7**	**32.2**	**36.6**	**19.1**	**33.1**	**37.2**	21.0
Retentive R-CNN [22]	✓	✓	31.5	**39.2**	8.3	32.1	39.2	10.5	32.9	**39.3**	13.8
CFA w/ fc [23]	✓	✓	31.8	**39.5**	8.8	32.2	**39.5**	10.4	33.2	**39.5**	14.3
CFA w/ cos [23]	✓	✓	32.0	**39.5**	**9.6**	**32.4**	**39.4**	11.3	33.4	**39.5**	15.1
CFA-DeFRCN [23]	✓	✓	**33.0**	38.9	**15.6**	**34.0**	39.0	**18.9**	**34.9**	39.0	**22.6**
NIFF-DeFRCN	✓	✗	**33.1**	38.9	**15.9**	**34.0**	39.0	**18.8**	**34.5**	39.0	**20.9**

Table 2. G-FSOD results on MS-COCO for $5, 10, 30$-shot settings. w/E denotes the ensemble-based evaluation protocol. w/B indicates whether the base data is available during novel finetuning. The best and second-best results for each evaluation protocol are color coded.

Feature(s) per class	10-Shot Inference					
	AP	bAP	nAP	AR	bAR	nAR
1	29.9	34.0	17.6	28.4	31.0	**20.7**
5	30.6	34.9	**17.7**	**28.8**	**31.5**	**20.8**
10	**30.7**	**35.0**	**17.8**	**28.8**	**31.5**	**20.8**
30	**30.8**	**35.1**	**17.8**	**28.8**	**31.5**	**20.8**
10 (fixed)	25.9	28.9	16.9	25.5	27.1	20.6
10 (10 sampled cls)	30.5	34.8	**17.7**	**28.7**	**31.4**	20.6

Table 3. Effect of the number of generated features per class and fixed samples on the forgetting and the detection performance without any added regularization.

tuning setting ($N = K = 10$-shot), the best overall results are achieved. Further increases yield almost similar results so we opt to set $N = K$ consistently throughout our experiments on MS-COCO and PASCAL-VOC. Next, in row (5), we generate 10 features per class by sampling only once at the start of the training and fix them throughout the novel training. We notice that the performance drops significantly due to the limited diversity of generated base features, emphasizing the importance of sampling the features in each iteration. In the final row, we randomly sample a random subset of the base classes $\mathcal{C}_s < |\mathcal{C}_b|$, but still generate $N = 10$ features for each. We choose $\mathcal{C}_s = 10$. Compared to generating features for all the base classes (row 3), we notice a slight drop in performance, highlighting the importance of a class-balanced sampling scheme. Hence, we opt to generate $N = K$ features per class for all the base classes in each iteration to achieve the best overall AP.

4.2. Main Comparisons

We compare our method (NIFF) against state-of-the art G-FSOD [22, 23] and FSOD [16, 17, 19] models on MS-COCO and PASCAL-VOC benchmarks. We opt to apply our approach on top of the recent state-of-the-art transfer learning based approach DeFRCN [19]. We denote our model by *NIFF-DeFRCN*.

Methods / Shots	w/B	5-Shot			10-Shot			30-Shot		
		AP	bAP	nAP	AP	bAP	nAP	AP	bAP	nAP
DeFRCN [19]	✓	28.7	33.1	15.3	30.6	34.6	**18.6**	31.6	34.7	**22.5**
DeFRCN	✗	23.7	26.3	**15.6**	18.2	18.5	17.4	16.3	16.3	21.4
DeFRCN w/ DA	✗	22.6	25.0	15.3	26.4	29.2	17.9	24.2	25.0	**21.8**
DeFRCN w/ LOD	✗	29.0	34.1	13.4	27.0	30.7	16.1	29.8	33.2	19.7
DeFRCN w/ MAS	✗	31.0	**36.8**	13.5	31.5	**36.8**	15.3	32.6	36.6	20.4
DeFRCN w/ EWC	✗	**31.1**	**37.1**	13.4	**31.8**	**36.9**	16.6	**33.0**	**37.3**	20.1
DeFRCN	$✓_F$	24.2	27.6	13.6	25.8	28.9	16.6	26.6	29.0	19.7
DeFRCN + CFA	$✓_F$	26.0	29.9	13.7	27.7	31.4	16.6	28.6	31.5	19.9
DeFRCN w/ KD	$✓_F$	25.3	28.8	14.3	27.0	30.2	17.4	27.9	30.3	20.5
DeFRCN w/ KD + CFA	$✓_F$	28.4	33.3	14.1	30.5	34.9	17.1	31.3	35.0	20.3
NIFF-DeFRCN	✗	**31.3**	36.3	**15.7**	**32.2**	36.6	**19.1**	**33.1**	**37.2**	21.0

Table 4. G-FSOD results for various baselines on top of DeFRCN [19] on MS-COCO for $5, 10, 30$-shot settings. $✓_F$ denotes finetuning with offline saved base RoI features. w/B indicates whether the base data is available during novel finetuning. KD denotes the proposed knowledge distillation approach.

Results on MS-COCO. In Tab. 2, we show the results on MS-COCO. (w/B) denotes whether base data is used. To show the impact of removing the base data on G-FSOD, we reevaluate our baseline DeFRCN [19] without any base data. We notice that both the base and novel performance drop across all shots. This shows the importance of the base data in the knowledge transfer to new tasks. By applying NIFF, we show that we are consistently able to boost the base performance across all settings yielding higher overall AP performance. Moreover, we evaluate our model using the ensemble evaluation protocol in Retentive R-CNN [22] and outperform the other approaches despite the absence of base data (with an 0.4AP less in 30-shot setting). It is essen-

Methods / Shots	w/E	w/B	All Set 1					All Set 2					All Set 3				
			1	2	3	5	10	1	2	3	5	10	1	2	3	5	10
FRCN-ft-full [16]	✗	✓	55.4	57.1	56.8	60.1	60.9	50.1	53.7	53.6	55.9	55.5	58.5	59.1	58.7	61.8	60.8
TFA w/ fc [16]	✗	✓	69.3	66.9	70.3	73.4	73.2	64.7	66.3	67.7	68.3	68.7	67.8	68.9	70.8	72.3	72.2
TFA w/ cos [16]	✗	✓	69.7	68.2	70.5	73.4	72.8	65.5	65.0	67.7	68.0	68.6	67.9	68.6	71.0	72.5	72.4
MPSR [17]	✗	✓	56.8	60.4	62.8	66.1	69.0	53.1	57.6	62.8	64.2	66.3	55.2	59.8	62.7	66.9	67.7
DeFRCN [19]	✗	✓	73.1	73.2	73.7	75.1	**74.4**	68.6	**69.8**	71.0	72.5	71.5	72.5	73.5	72.7	74.1	73.9
Meta R-CNN [10]	✗	✓	17.5	30.5	36.2	49.3	55.6	19.4	33.2	34.8	44.4	53.9	20.3	31.0	41.2	48.0	55.1
FSRW [31]	✗	✓	53.5	50.2	55.3	56.0	59.5	55.1	54.2	55.2	57.5	58.9	54.2	53.5	54.7	58.6	57.6
FsDetView [12]	✗	✓	36.4	40.3	40.1	50.0	55.3	36.3	43.7	41.6	45.8	54.1	37.0	39.5	40.7	50.7	54.8
CFA w/ fc [23]	✗	✓	69.5	68.2	69.8	73.5	74.3	66.0	66.9	69.2	70.1	71.1	67.7	69.0	70.9	72.6	73.5
CFA w/ cos [23]	✗	✓	69.1	69.8	71.9	73.6	73.9	64.8	66.5	68.3	69.5	70.5	67.7	69.7	71.9	73.0	73.5
CFA-DeFRCN [23]	✗	✓	**73.8**	**74.6**	**74.5**	**76.0**	**74.4**	**69.3**	**71.4**	**72.0**	**73.3**	**72.0**	**72.9**	**73.9**	**73.0**	**74.1**	**74.6**
DeFRCN	✗	✗	61.1	48.5	35.9	32.8	20.7	64.7	59.7	58.2	56.9	48.4	56.3	51.2	46.9	38.8	23.9
NIFF-DeFRCN	✗	✗	**75.6**	**76.5**	**76.7**	**77.4**	**76.9**	**70.0**	**71.4**	**73.9**	**74.4**	**74.0**	**74.4**	**75.8**	**76.2**	**76.6**	**76.7**
Retentive R-CNN [22]	✓	✓	71.3	72.3	72.1	74.0	74.6	66.8	68.4	70.2	70.7	71.5	69.0	70.9	72.3	73.9	74.1
CFA w/ fc [23]	✓	✓	70.3	69.5	71.0	74.4	74.9	67.0	68.0	70.2	70.8	71.5	69.1	70.1	71.6	73.3	74.7
CFA w/ cos [23]	✓	✓	71.4	71.8	73.3	74.9	75.0	66.8	68.4	70.4	71.1	71.9	**69.7**	71.2	72.6	74.0	74.7
CFA-DeFRCN [23]	✓	✓	**75.0**	**76.0**	**76.8**	**77.3**	**77.3**	**70.4**	**72.7**	**73.7**	**74.7**	**74.2**	**74.7**	**75.5**	**75.0**	**76.2**	**76.6**
NIFF-DeFRCN	✓	✗	**75.9**	**76.9**	**77.3**	**77.9**	**77.5**	**70.6**	**71.6**	**74.5**	**75.1**	**74.5**	**74.7**	**76.0**	**76.1**	**76.8**	**76.7**

Table 5. G-FSOD (AP50) results on PASCAL-VOC for 1, 2, 3, 5, 10-shot settings for all three splits are reported.

tial to note that this evaluation protocol requires keeping the base model parameters [22,23], increasing the overall memory footprint and inference time. Note that our approach NIFF-DeFRCN (w/o E and w/o B) outperforms Retentive R-CNN (w/E and w/B) in the overall AP.

Comparison against regularization-based continual learning and G-FSOD baselines. In Tab. 4, we compare our method with base-data-free and GFSOD baselines on top of DeFRCN [19]. The data-free baselines are drawn from regularization-based continual learning works, namely: pixel-level data augmentations (DA), EWC [30] and memory aware synapses (MAS) [32] with the computed FIM, and lifelong object detection (LOD) [33]. Since CFA [23] cannot be conducted in a data-free setting, we train DeFRCN and save base RoI features to later apply CFA during novel training. We also add two baselines: the proposed KD method on top of DeFRCN using saved base RoI features with and without CFA. The proposed method is able to consistently outperform both data-free and data-reliant baselines. We argue that the diversity of the forged features is what allows our method to surpass data-reliant baselines. Moreover, it is important to note that the stored features require 114.8MB, which is significantly more than the memory required by our generator (3.7MB) and stats (12.4MB). Similarly, EWC and MAS are memory intensive as the FIM requires around 200MB.

Results on PASCAL-VOC. We report the overall performance on PASCAL-VOC (AP50) in Tab. 5. We show that adopting NIFF achieves state-of-the-art results with and without the ensemble evaluation protocol. Due to the limited space, we report the novel performance (nAP50) results on PASCAL-VOC. Further ablation experiments on MS-COCO and experiments with multiple runs on both datasets in the Supplementary materials. Although the performance on novel classes is not our primary objective, NIFF-DeFRCN achieves competitive results on both datasets are shown in the majority of cases.

4.3. Memory & Computation Analysis

For 10-shot MS-COCO, the overall memory required is (Model = 195.16MB, Base Images = 148.8MB, Novel Images= 48.6MB). The Generator has 3.7MB, the stats have 12.42MB, so the proposed model saves 33.8% of the initial requirements. Computationally, DeFRCN and the generator require 133.46G and 943.94K FLOPS, respectively, implying that the computational overhead is negligible. DeFRCN G-FSOD training requires 104.5 mins, where our generator training and data generation additionally require 112 mins and 27 mins, respectively.

5. Conclusion

We propose NIFF, a data-free G-FSOD framework, which alleviates forgetting on the base images all without using any base data. Our main contribution is the design of a *standalone* generator that forges base *instance-level features* instead of images by aligning *class-wise statistics* in the RoI head. The generator has a negligible memory footprint ($\sim$ 4MB) which is two orders of magnitude lower than using base images for finetuning. Replaying the forged features during novel finetuning, along with careful design choices in the training pipeline, results in state-of-the-art overall AP performance. In the future, we plan to apply our approach on the recent state-of-the-art transformer-based FSOD models. Furthermore, we encourage future works to extend G-FSOD in meta-learning paradigms as their base performance drops significantly. Finally, we hope that our proposed approach of using a standalone generator in the feature space sheds light on works in areas other than object detection.

Acknowledgment This research was supported by the German Federal Ministry for Economic Affairs and Climate Action (BMWK) within the project FabOS under grant number 01MK20010I. The authors would like to thank Mohamed Sayed from UCL for the helpful and informative discussions.

References

[1] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In IEEE Conference on Computer Vision and Pattern Recognition, pages 580–587, 2014. 1

[2] Ross Girshick. Fast R-CNN. In IEEE International Conference on Computer Vision, pages 1440–1448, 2015. 1

[3] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In Advances in Neural Information Processing Systems, pages 91–99, 2015. 1, 5

[4] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 6154–6162, 2018. 1

[5] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. SSD: Single shot multibox detector. In European Conference on Computer Vision, pages 21–37, 2016. 1

[6] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In IEEE Conference on Computer Vision and Pattern Recognition, pages 779–788, 2016. 1

[7] Joseph Redmon and Ali Farhadi. YOLO9000: better, faster, stronger. In IEEE Conference on Computer Vision and Pattern Recognition, pages 7263–7271, 2017. 1

[8] Tsung-Yi Lin, Priyal Goyal, Ross Girshick, Kaiming He, and Piotr Dollar. Focal loss for dense object detection. IEEE Transactions on Pattern Analysis and Machine Intelligence, 42(2):318–327, 2018. 1

[9] Simone Antonelli, Danilo Avola, Luigi Cinque, Donato Crisostomi, Gian Luca Foresti, Fabio Galasso, Marco Raoul Marini, Alessio Mecca, and Daniele Pannone. Few-shot object detection: A survey. ACM Comput. Surv., 54(11s), sep 2022. 1

[10] 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 IEEE International Conference on Computer Vision, pages 9577–9586, 2019. 1, 2, 7, 8

[11] Qi Fan, Wei Zhuo, and Yu-Wing Tai. Few-shot object detection with attention-rpn and multi-relation detector. In IEEE Conference on Computer Vision and Pattern Recognition, pages 4012–4021, 2020. 1, 2

[12] Yang Xiao and Renaud Marlet. Few-shot object detection and viewpoint estimation for objects in the wild. In European Conference on Computer Vision, pages 192–210, 2020. 1, 2, 7, 8

[13] 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 IEEE Conference on Computer Vision and Pattern Recognition, pages 7359–7368, 2021. 1, 2

[14] Tung-I Chen, Yueh-Cheng Liu, Hung-Ting Su, Yu-Cheng Chang, Yu-Hsiang Lin, Jia-Fong Yeh, Wen-Chin Chen, and Winston Hsu. Dual-awareness attention for few-shot object detection. IEEE Transactions on Multimedia, pages 1–1, 2021. 1, 2

[15] Gongjie Zhang, Zhipeng Luo, Kaiwen Cui, and Shijian Lu. Meta-DETR: Few-shot object detection via unified image-level meta-learning. CoRR, abs/2103.11731, 2021. 1, 2

[16] Xin Wang, Thomas E. Huang, Trevor Darrell, Joseph E. Gonzalez, and Fisher Yu. Frustratingly simple few-shot object detection. In International Conference on Machine Learning, pages 9919–9928, 2020. 1, 2, 6, 7, 8

[17] Jiaxi Wu, Songtao Liu, Di Huang, and Yunhong Wang. Multi-scale positive sample refinement for few-shot object detection. In European Conference on Computer Vision, pages 456–472, 2020. 1, 2, 7, 8

[18] Bo Sun, Banghuai Li, Shengcai Cai, Ye Yuan, and Chi Zhang. FSCE: few-shot object detection via contrastive proposal encoding. In IEEE Conference on Computer Vision and Pattern Recognition, pages 7352–7362, 2021. 1

[19] Limeng Qiao, Yuxuan Zhao, Zhiyuan Li, Xi Qiu, Jianan Wu, and Chi Zhang. DeFRCN: Decoupled faster R-CNN for few-shot object detection. In IEEE International Conference on Computer Vision, 2021. 1, 2, 6, 7, 8

[20] Hongxu Yin, Pavlo Molchanov, Jose M. Alvarez, Zhizhong Li, Arun Mallya, Derek Hoiem, Niraj K. Jha, and Jan Kautz. Dreaming to distill: Data-free knowledge transfer via deepinversion. In 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 8712–8721, 2020. 2, 3

[21] James Smith, Yen-Chang Hsu, Jonathan Balloch, Yilin Shen, Hongxia Jin, and Zsolt Kira. Always be dreaming: A new approach for data-free class-incremental learning. In 2021 IEEE/CVF International Conference on Computer Vision (ICCV), pages 9354–9364, 2021. 2, 3, 5, 6

[22] Zhibo Fan, Yuchen Ma, Zeming Li, and Jian Sun. Generalized few-shot object detection without forgetting. In IEEE Conference on Computer Vision and Pattern Recognition, pages 4527–4536, 2021. 1, 3, 5, 6, 7, 8

[23] Karim Guirguis, Ahmed Hendawy, George Eskandar, Mohamed Abdelsamad, Matthias Kayser, and Jürgen Beyerer. Cfa: Constraint-based finetuning approach for generalized few-shot object detection. In 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pages 4038–4048, 2022. 1, 3, 5, 6, 7, 8

[24] A. Mordvintsev, Christopher Olah, and Mike Tyka. Inceptionism: Going deeper into neural networks. 2015. 2, 3

[25] Akshay Chawla, Hongxu Yin, Pavlo Molchanov, and Jose Alvarez. Data-free knowledge distillation for object detection. In 2021 IEEE Winter Conference on Applications of Computer Vision (WACV), pages 3288–3297, 2021. 2, 3, 4

[26] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating deep network training by reducing internal covariate shift. In International Conference on Machine Learning, pages 448–456, 2015. 2

[27] 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 European Conference on Computer Vision, pages 740–755. Springer, 2014. 2, 6

[28] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (VOC) challenge. International Journal of Computer Vision, 88(2):303–338, 2010. 2, 6

[29] Juan-Manuel Perez-Rua, Xiatian Zhu, Timothy M. Hospedales, and Tao Xiang. Incremental few-shot object detection. In IEEE Conference on Computer Vision and Pattern Recognition, pages 13843–13852, 2020. 2, 7

[30] James Kirkpatrick, Razvan Pascanu, Neil C. Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran

Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. Proceedings of the National Academy of Sciences, 114, 2016. 6, 8

[31] Bingyi Kang, Zhuang Liu, Xin Wang, Fisher Yu, Jiashi Feng, and Trevor Darrell. Few-shot object detection via feature reweighting. In IEEE International Conference on Computer Vision, pages 8419–8428, 2018. 7, 8

[32] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In ECCV, pages 144–161, 2018. 8

[33] Wang Zhou, Shiyu Chang, Norma E. Sosa, Hendrik F. Hamann, and David D. Cox. Lifelong object detection. volume abs/2009.01129, 2020. 8