Few-shot Camouflage Animal Detection and Segmentation

Thanh-Danh Nguyen[1,4†], Anh-Khoa Nguyen Vu[1,4†], Nhat-Duy Nguyen[1,4†], Vinh-Tiep Nguyen[1,4], Thanh Duc Ngo[1,4], Thanh-Toan Do[5], Minh-Triet Tran[2,3,4], Tam V. Nguyen[6*]

[1]University of Information Technology, Ho Chi Minh City, Vietnam.
[2]University of Science, Ho Chi Minh City, Vietnam.
[3]John von Neumann Institute, VNU-HCM, Vietnam.
[4]Vietnam National University, Ho Chi Minh City, Vietnam.
[5]Monash University, Clayton, VIC 3800, Australia.
[6*]University of Dayton, Dayton, OH 45469, United States.

*Corresponding author(s). E-mail(s): tamnguyen@udayton.edu;
Contributing authors: danhnt@uit.edu.vn; khoanva@uit.edu.vn; duynn@uit.edu.vn; tiepnv@uit.edu.vn; thanhnd@uit.edu.vn; toan.do@monash.edu; tmtriet@fit.hcmus.edu.vn;
†These authors contributed equally to this work.

Abstract

Camouflaged object detection and segmentation is a new and challenging research topic in computer vision. There is a serious issue of lacking data of camouflaged objects such as camouflaged animals in natural scenes. In this paper, we address the problem of few-shot learning for camouflaged object detection and segmentation. To this end, we first collect a new dataset, CAMO-FS, for the benchmark. We then propose a novel method to efficiently detect and segment the camouflaged objects in the images. In particular, we introduce the instance triplet loss and the instance memory storage. The extensive experiments demonstrated that our proposed method achieves state-of-the-art performance on the newly collected dataset.

Keywords: Camouflaged Instance, Camouflaged Animal, Few-shot Learning, Object Detection, Instance Segmentation

1 Introduction

Camouflage is a defense mechanism that animals use to conceal their appearance by blending in with their environment [1]. Autonomously detecting camouflaged animals is helpful in various applications, e.g., search-and-rescue missions [2]; wild species discovery and preservation activities [2]; and media forensics (manipulated image/video detection and segmentation [3]). Although image segmentation methods have been proposed for a long time, general detectors cannot deal with camouflaged animals [4–7]. The detectors initially developed for camouflage detection [8–15], which use handcrafted low-level features, are effective only for images with a simple and uniform background. More recently developed deep learning-based detectors [2, 16] for camouflaged object segmentation rely on large-scale data.

With the approach of few-shot learning, we can perform machine learning tasks with given limited data. A few-shot method requires two stages of processing: (1) one base phase training for the model to gain concept knowledge of general domains with abundant data, and then (2) performing a novel phase which can do the specific task on few-shot data. In the case of the camouflaged object detection and segmentation task, we leverage few-shot learning in the concept of camouflaged animals which is rare and hard to find in the wild. Thus, with limited data on camouflaged objects, the models can still well handle the given tasks. However, none of the aforementioned publications targets few-shot camouflaged object detection and segmentation despite its practical applications. The task of segmentation supports better to identify camouflaged objects in terms of specifying which pixels in the images conceal the objects in comparison to classification and detection. In fact, the research on camouflaged animals suffers due to the lack of data. There are not many object classes with rare instances captured in photos. Therefore, in this paper, we would like to address few-shot learning with camouflaged animals.

Our contributions in this work are two-fold:

- First, we build a new benchmark dataset, CAMO-FS, which is among the first datasets to support few-shot detection and instance segmentation on camouflaged instances in nature.
- Second, we propose FS-CDIS, a framework to efficiently detect and segment camouflaged instances given a small shot of training data for novel classes utilizing an instance triplet loss and memory storage.

The remainder of this paper is organized as follows. Section 2 summarizes related work. Next, Section 3 introduces the newly constructed CAMO-FS dataset and presents our proposed framework for few-shot camouflaged object detection and segmentation. Section 4 presents the results of our evaluation of baselines on the newly constructed dataset. Finally, Section 5 summarizes the key points and mentions future work.

2 Related Work

2.1 Camouflage Research

Given any region (i.e. bounding boxes or polygon masks) presented for an object of interest (i.e. animals or artificial objects) in an image and then they tend to be classified as background, contents in that region can be qualified as camouflaged objects. Thus, a camouflaged object is defined as a set of bounding boxes or camouflaged pixels in an image without any further detailed information such as the number of objects or the semantic meaning [2]. Although tasks related to camouflaged animals are performed in a wide range of applications, this research field has not been well explored in the literature, especially few-shot learning which is practically suitable to the context of scare data as camouflaged animals.

Binary camouflage segmentation. Prior to the advancement of deep neural networks, most of the work exploits identical regions between camouflaged regions and the background by handcrafted or low-level features, specifically based on external characteristics (e.g., color, shape, orientation, and brightness). Particularly, early camouflage detection works had attention on the foreground region even when some of its texture was similar to the background [8–10, 17]. The foreground was distinguishable from the background via simple features, such as color, intensity, shape, orientation, and edge [17–21]. A few methods [11–15, 22] based on handcrafted low-level features have been proposed for tackling the problem of camouflage detection. However, they are effective only for images with a simple and uniform background. Thus, their performances are unsatisfactory in camouflaged object segmentation due to the substantial similarity between the foreground and the background.

Until now, the convention of binary prefers binary ground truth camouflaged object datasets [2, 16, 23]. Existing methods for camouflaged objects [2, 16, 24–29] based on binary ground truth are considered as the binary camouflage segmentation. For example, Le *et al.* [2] proposed an end-to-end Anabranch Network, dubbed ANet which includes two streams of classification and segmentation. The outputs of both streams are fused to improve the segmentation performance of camouflaged objects. This proposed network was also flexibly applied to any fully convolutional networks. Similarly, motivated by the way of hunting strategies of predators, Fan *et al.* [16] designed Search Identification Network (SINet) with two main modules to simulate this hunting behavior, namely a search module searching for targets and an identification module identifying the existence of targets then catching them. Yan *et al.* [27] recently introduced MirrorNet, a dual-stream network comprising a mainstream and a mirror stream. This mirror stream aimed to capture instinct information by horizontally flipping camouflaged objects to break their camouflaged nature and make them more distinguishable. Zhu *et al.* [28] presented the TINet, which interactively refines multi-level texture and segmentation features and thereby gradually enhances the segmentation of camouflaged objects. Lv *et al.* [29] simultaneously worked on ranking and localization to well-present camouflaged objects. As a result, they formed a triplet task with localizing, segmenting, and ranking the camouflaged objects. Besides, the authors also introduced the NC4K dataset for camouflaged segmentation. Such methods reveal the presence of the camouflaged objects with the high level of bounding

boxes and contain corresponding pixel-wise ground truth belonging to camouflage. Further understanding of the camouflage level may help us to give comparative analyses, finding evidence for links between camouflage and other defensive strategies with aspects of habitat and life-history [30].

Camouflage instance segmentation. Although several works have been proposed, there is still a difficulty in efficiently exploring the information of camouflage animals, especially at the instance level with more challenging detailed masks. Therefore, for ease of training methods with the challenging task of camouflaged instance segmentation, Le *et al.* [31] introduced a framework with several state-of-the-art methods and proposed a tool with user interactive cues to tune the segmentation mask on a website. Realizing that the semantic level is not detailed enough, Le *et al.* [32] introduced a camouflage fusion learning (CFL) to utilize the strength of different instance segmentation methods by fusing various models via learning image contexts.

Camouflage datasets. CamouflagedAnimals [33] and CHAMELEON [23] were the first two camouflage datasets with mask annotations. The two datasets do not contain enough images to train deep learning methods. Le *et al.* [2] created the CAMO dataset, the first camouflage dataset with more than $1,000$ annotated images. It contains $1,250$ annotated images, which is a limited number of samples to train and evaluate deep learning methods. Then, Fan *et al.* [16] collected the COD dataset, which comprises $10,000$ images (both camouflage and non-camouflage) divided into 5 meta-categories. However, they annotated only $5,066$ camouflage images. Lamdouar *et al.* [34] recently developed the MoCA dataset for the camouflage object detection task; it contains only bounding box ground truths. Hence, these datasets limit their annotations at binary ground truth datasets which have a shortage of intensive annotations for multi-task camouflage problems. CAMO++ [32] is different from the above-mentioned dataset, CAMO++ provides a benchmark for camouflaged instance segmentation with more comprehensive annotations and diverse meta-categories of 10. The dataset comprises $5,500$ images with superiority over other datasets on instances including $32,756$ instances for both camo and non-camo objects.

2.2 Few-shot Learning

Few-shot object detection (FSOD). When having some available samples of given classes with their corresponding bounding boxes, FSOD aims to learn from these limited data in order to help models adapt to the new classes. To date, several works [35–38] have been proposed to deal with FSOD. Early works [36, 38] mainly prefer to overcome the difficulties of the data scarcity of FSOD via meta-learning approaches by combining supportive information from meta-based streams with their main streams. Particularly, Bingyi [36] proposed a Feature Reweighting framework that leverages the free-proposal approach of a well-known one-stage framework such as YOLO [39] to boost FSOD performance. The network integrated a meta-model that aims to generate reweighting vectors from support samples for highlighting the attention to features from the YOLO network. Conversely, Meta RCNN [38] based on the two-stage proposal approach as Mask RCNN [40] and fed available annotations such as bounding boxes and segmented masks to train a meta-network called Predictor-head Remodeling Network for inferring attention features. Fan *et al.* [35] recently proposed to take

Table 1: Statistics of camouflage datasets (without non-camo images).

Dataset	Year	Publication	Type	#Annot. Camo. Img.	#Meta-Cat.	#Obj. Cat.	#Ins. or #Obj. per Img.	Bbox. GT	Obj. Mask GT	Ins. Mask GT	Few-shot
CamouflagedAnimals [33]	2016	ECCV	Video	181	-	6	1.238	×	✓	✓	×
MoCA [34]	2020	ACCV	Video	7,617	-	67	1.000	✓	×	×	×
CHAMELEON [23]	2018	-	Image	76	-	-	1.000	×	✓	×	×
CAMO [2]	2019	CVIU	Image	1,250	2	8	1.000	×	✓	×	×
COD [16]	2020	CVPR	Image	5,066	5	69	1.171	✓	✓	✓	×
CAMO++ [32]	2022	TIP	Image	2,695	10	47	1.171	✓	✓	✓	×
CAMO-FS (Ours)	2023	-	Image	2,858	10	47	1.172	✓	✓	✓	✓

advantage of support images from a massive FSOD dataset to generate significant results combined with their proposed network called Attention-RPN, Multi-Relation Detectors. The Attention-RPN directed the trained model to look at the image for the task of object detection. Differently, Wang *et al.* [37] simply adopted Faster RCNN with two-stage finetuning to transfer massive knowledge from abundant data in the base model to fine-tune the novel one by freezing the whole network except for the fully connected layer for object classification. Through this simple straightforward mechanism, this model significantly improved few-shot performance without a complex pipeline of training the model. Further, such works [41–45] presented advanced methods by applying class max-margin, multiple scale proposals, or feature alignment in FSOD. Other ones were based on transformed inputs [46, 47], transformer approaches [48, 49], contrastive method [50], or kernels design [51]. Other methods [37, 52–54] relied only on query images to deal with FSOD via extra text data [54], unlabeled image [55], generated samples [53], gradient scaling [52].

Few-shot object segmentation (FSOS). Recently, the field of few-shot segmentation gained attention from the community. As mentioned above, the first work Meta RCNN originated from Mask RCNN, therefore, Meta RCNN simultaneously performed detection and segmentation. Liu *et al.* [56] utilized a cross-reference network for generic image segmentation. The authors proposed a cross-reference mechanism and a mask refinement module to specifically support the task of segmentation. Before, Dong *et al.* [57] proposed a prototype learning component in a framework of semantic segmentation that learned to take discriminative information from features to help segment objects better. Also, Wang *et al.* [58] introduced a prototype align method that learns class-specific prototype representations from a few image samples to perform segmentation over the query images. Lately, Liu *et al.* proposed a dynamic prototype convolution network to address few-shot semantic segmentation. The work of [59] proposed context-aware prototype learning. [60] introduced generative models approach for this task. Recently, Nguyen *et al.* [61] came up with iFS-RCNN, an instance segmenter via an incremental approach. Gao *et al.* [62] proposed the DCFS framework, an effective decoupling classifier that boosted the performance of object detection and segmentation heads. Han *et al.* [63] suggested a reference twice transformer-based framework (RefT) to enhance features in segmentation tasks. Also in the transformer approach, Wang *et al.* [64] introduced DTN to directly segment the target object instances from arbitrary categories given reference images. In common, these aforementioned methods focus on generic objects.

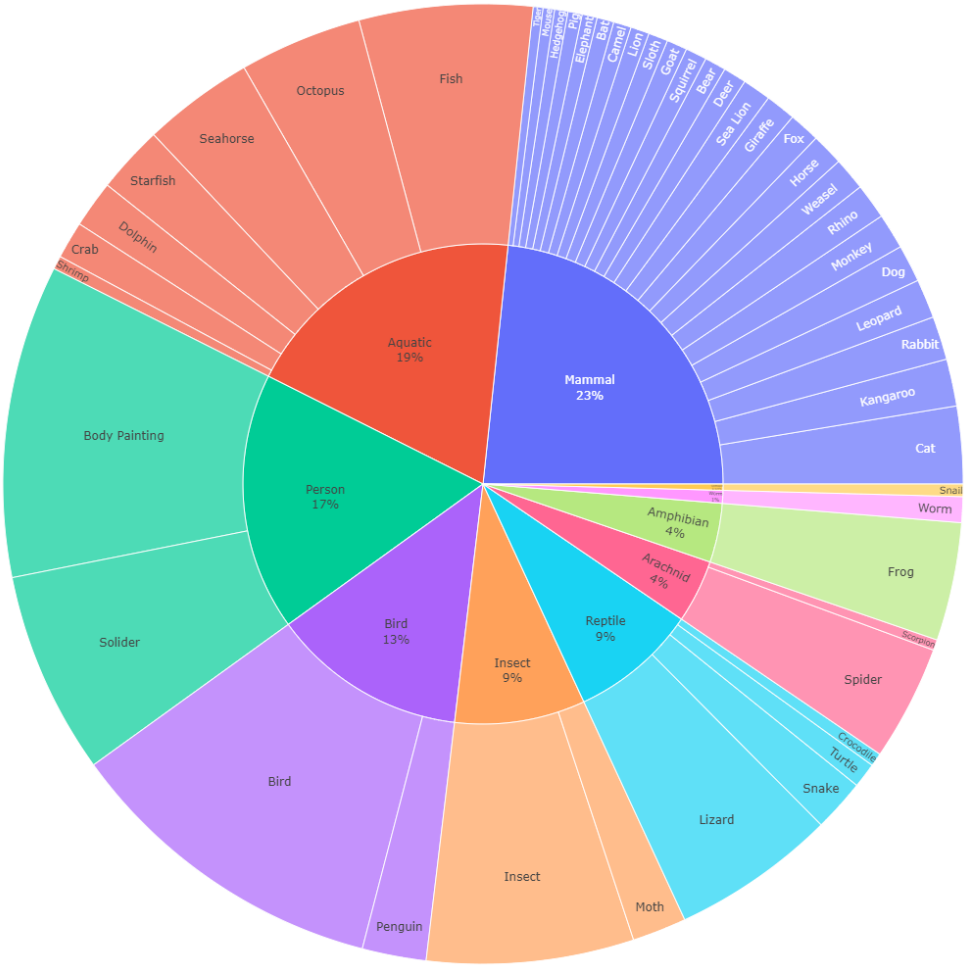


Fig. 1: Hierarchical taxonomic structure of our CAMO-FS dataset and the corresponding distribution of camouflaged coarse-grained classes. Best viewed online in color and zoomed in.

3 Proposed Method

3.1 CAMO-FS Benchmark Dataset

Camouflaged data tends to be more difficult to collect in the real world rather than non-camouflaged ones. Generating intensive annotations with multi-task or hierarchical labels for camouflaged objects is also costly and complicated, especially with the pixel level as polygon masks. Particularly, the visual characteristics of a camouflaged object are extremely identical to the background. The external appearances (i.e. the intensity, color, and textures) are close to their surrounding environment, the boundary

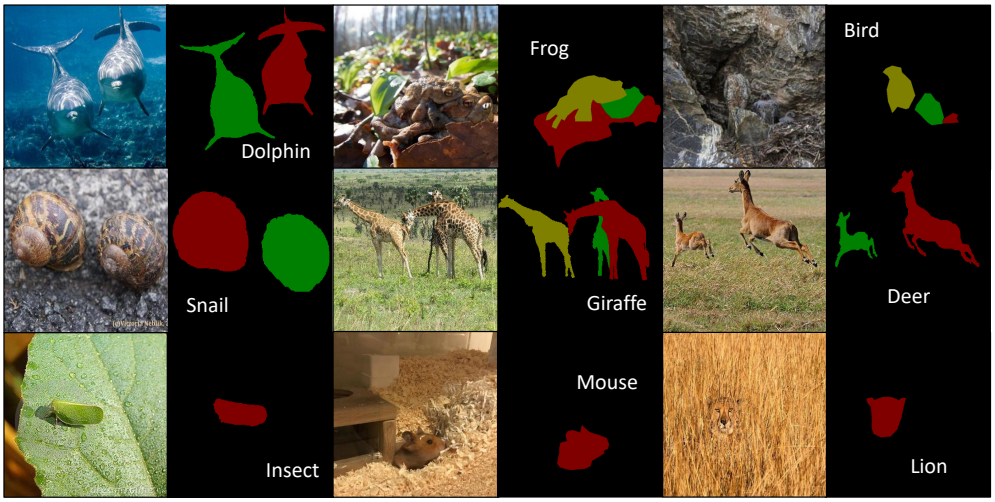
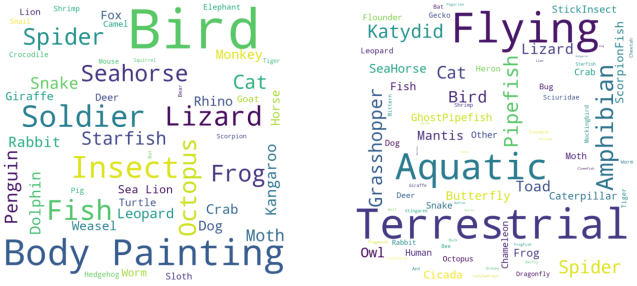


Fig. 2: Exemplary images with instance-level mask annotations from our proposed CAMO-FS dataset.

between camouflaged objects and the background or other identical-type camouflaged objects in case of being nearly or partly overlapped. Thus, it is really tough to provide the concurrence between annotators due to ambiguity in verifying camouflaged regions blended in surroundings. For ease of data preparation such as collections and annotations, one of the most common way is that inherits existing camouflaged datasets and CAMO++ [32] is our selected dataset since it is a high-diversity dataset with a variety of camouflaged object categories. Furthermore, the key to few-shot learning lies in the generalization ability of the pertinent model when presented with a few available samples. The context of camouflaged objects inherently matches this understanding because the number of camouflaged images is often scarce in practice.

CAMO++ Dataset. CAMO++ generally contains camouflaged and non-camouflaged images with a total of 5,500 images corresponding to 32,756 instances [32]. The dataset contains 93 fine-grained classes assigned to 13 coarse-grained classes. However, in the case of camouflaged objects, there are 47 fine-grained classes designed with a hierarchical structure and assigned into 10 coarse-grained classes. In detail, CAMO++ contributes 2,695 camouflage images including 1,250 existing camouflage images in the previous CAMO dataset with 1,450 newly collected camouflage images for CAMO++. In this scope of our paper, 2,800 remaining non-camouflage images are ignored. CAMO++ especially provides common ground truths such as bounding boxes, object masks, and instance masks which are suitable for many tasks of camouflage research.

CAMO-FS Dataset. We leverage the available CAMO++ to build our CAMO-FS dataset. In this way, we inherit the biology taxonomic and vision taxonomic structure of CAMO++ which helps us to reduce the burden of data collection. Table 1 provides an overview of previous works done on camouflage, which is mentioned in the related work, and our proposed CAMO-FS in terms of main characteristics. We exploit the



(a) Our CAMO-FS (b) COD [16]

Fig. 3: Word cloud of class-distribution of camouflaged instances.

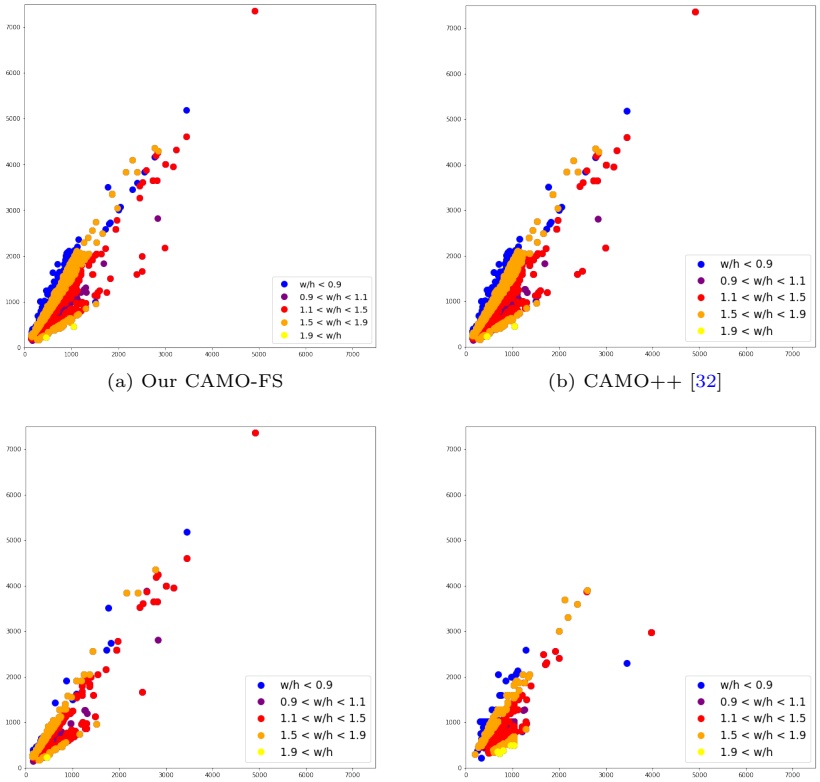
Table 2: Extra collected number of images and instances in CAMO-FS dataset.

Classes	Bat	Bear	Camel	Dolphin	Elephant	Horse	Kangaroo	Monkey	Penguin	Rhino	Squirrel	Total
#images	12	14	14	13	14	16	22	16	11	14	17	163
#instances	12	14	15	19	14	17	25	20	14	14	17	181

diversity of CAMO++ by its 10 meta-categories to build up the few-shot concept for instance segmentation. To this end, our CAMO-FS not only keeps a good ratio of instances per image of 1.172 but also contributes as the very first dataset specific for few-shot research on camouflaged animals. Note that the large amount of images in some datasets does not mean they are all camouflaged images. Figure 3 illustrates the class-distribution of our CAMO-FS dataset and COD [16]. Our CAMO-FS shares similar categorical diversity with the original CAMO++ [32].

However, imbalanced data and a shortage of the number of images of some classes inherently exist in CAMO++ posing problems of evaluation for few-shot tasks. Particularly, there are 11 classes (*e.g. Camel, Dolphin, Elephant, Horse, Kangaroo, Monkey, Penguin, Bat, Bear, Squirrel and Rhino*) even having a shortage of images that are needed to train a few-shot model. Hence, we hardly perform training or testing on these classes. As a result, we collect more data for these classes with 163 total images corresponding to 181 instances (an average of 15-16 instances per class). We also remove images with mistakes in the original dataset. The statistics of collected data are shown in Table 2. By gathering more camouflaged animals and combining them with the CAMO++ dataset, we conduct our CAMO-FS dataset for few-shot camouflaged animal detection and segmentation with 2,858 total images corresponding to 3,342 instances. Figure 1 shows the vision taxonomic structure of coarse-grained and corresponding fine-grained classes and illustrates the ratios of 10 coarse-grained classes in our proposed CAMO-FS dataset. Figure 2 visualizes the exemplary images with mask annotations from our proposed CAMO-FS.

In Table 3, we report the aggregated number of instances per image. The number of instances per image ranges from 1 to 25 and commonly falls into 1, then 2 and 3 while the remaining is beyond 3 instances. As can be seen, the number of images that



(a) Our CAMO-FS (b) CAMO++ [32]

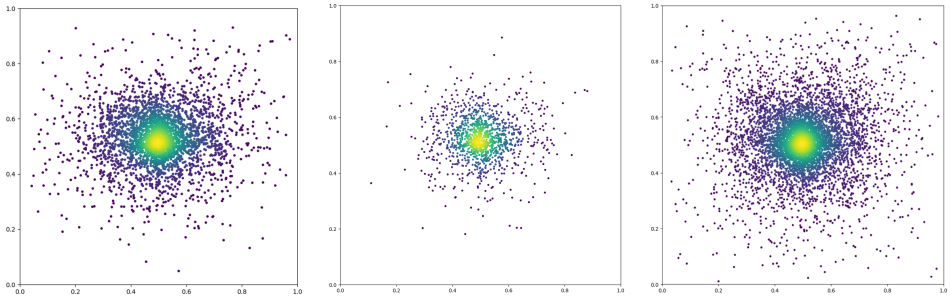
(c) CAMO [2] (d) COD [16]

Fig. 4: Distribution of camouflage image resolution. Best viewed online in color and zoomed in.

No. of instances	Ratio (%)	#Images
1	90.5	2581
2	1.05	190
3	1.79	51
3+	6.66	30

Table 3: CAMO-FS dataset instances per image distribution.

contain 1 to 3 instances takes up a large proportion of the entire dataset. This also illustrates the problem of data imbalance between the number of instances and the ratio of images in the dataset, which reflects a problem that the presence of camouflaged animals captured in photos is often limited, i.e. mostly one animal per image. Additionally, although being claimed in [32] that camouflaged objects in CAMO++ were localized over the entire image, after removing non-camouflage objects and adding new camouflaged images, we have the distributions of object centers in normalized image coordinates over all images in the CAMO-FS dataset as in Figure 5-a. This means



(a) Our CAMO-FS (b) CAMO [2] (c) COD [16]

Fig. 5: Instance center bias camouflaged datasets. Best viewed online with color and zoomed in.

camouflaged animals tend to be located in the center of images. Indeed, to capture images of camouflaged animals in the wild, photographers need to carefully focus on the animals, which leads to the central layout of collected images. Also in Figure 5, we illustrate the center bias of camouflaged images in other CAMO [2] and COD [16] datasets for better visual comparison. In Figure 4, we present the image resolution among camouflage datasets. As we only consider camouflaged images of CAMO++ [32] and COD [16], the density of our CAMO-FS is slightly higher than CAMO++ as a result of our extra collection of images presented in Table 2. In comparison with the previous COD [16] and CAMO [2], our CAMO-FS image resolution distribution is more satisfying in diversity.

To effectively create the data for the few-shot problem, we get M instances from the CAMO-FS dataset to create training sets (in our setup, $M = 5$) and use the remaining instances for testing. We only remove some objects of the higher-level training set if it exists to create the other few-shot settings. For example, we get all elements to generate 5-shot training data and discard 2 in 5 objects to make a 3-shot one. In this way, the 5-shot benchmark contains objects of the 3-shot dataset and the 3-shot setting contains the objects of the 2-shot one.

To the best of our knowledge, this is among the first works to address few-shot camouflaged instance segmentation and detection. Given the lack of a large-scale dataset for training and testing purposes on camouflaged animal issues, we build a benchmark for the task of few-shot camouflaged instance segmentation and detection.

3.2 General Framework

Few-shot instance segmentation formulation. In few-shot learning, we have one set of base classes denoted C_{base} with a large amount of available training data, and one disjoint set of novel classes denoted C_{novel} containing a small amount of training data. This amount is small to a few samples. The ultimate goal is to train a model to predict well on the novel classes $C_{test} = C_{novel}$ [65, 66] or on both base and novel data $C_{test} = C_{base} \cup C_{novel}$ [67]. In few-shot classification, this work [66] introduces the method of episodic training. The method sets up a series of episodes $E_i = (I_q, S_i)$

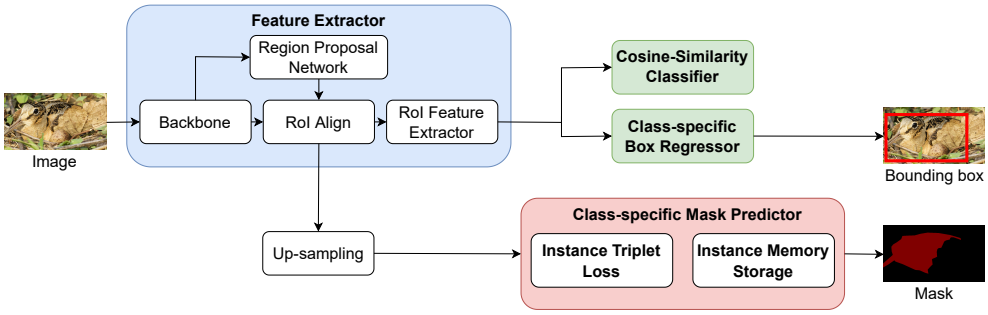


Fig. 6: Our general FS-CDIS framework for Few-Shot Camouflaged Detection and Instance Segmentation.

where S_i is a support set that contains N classes from $C_{train} = C_{novel} \cup C_{base}$ along with K examples per class (so-called N-way K-shot). A network is then trained to classify an input image I_q, termed query image, out of the classes in S_i. The key idea is that solving a different classification task for each episode leads to better generalization and results on C_{novel}. The extended versions of this method are FSOD [36] and FSIS [38, 68]. Those proposals consider all objects in an image as queries and they have a single support set per image instead of per query. However, there exist challenges in FSIS which are not only classification tasks on the query objects but also how to determine their localization and segmentation. Use an image I_q to query, FSIS returns labels y_i, bounding boxes b_i, and segmentation masks M_i for all objects in I_q that belong to the set of C_{test}.

General framework. Originated from TFA [37] which uses Faster R-CNN [69], MTFA [70] employs a mask prediction branch to return the pixel-wise mask for the segmentation task. In this work, we leverage the architecture of MTFA model [70] based on Mask R-CNN [40] which is a two-stage training and fine-tuning mechanism. We train the first stage of the framework on 80 classes from the COCO dataset. This stage results in the base model weights for the second stage of novel fine-tuning. In the fine-tuning stage, we apply the few-shot technique to learn the novel concepts of camouflaged instances in our proposed CAMO-FS dataset.

Similar to Mask R-CNN, the input images are fed into a feature extractor F consisting of backbone B, RoI Align, RoI feature extractor modules, and a region proposal network. There are three heads specifying three tasks that this scheme supports: a classification head C, a box regression head R, and a new attached mask prediction head M. In the first stage, the network is trained on the base classes C_{base}. Then in the second stage, we froze the backbone network B of the feature extractor F and only perform training on the prediction heads. Thus, only RoI classifier C, box regressor R, and mask predictor M are fine-tuned in the second stage. In Figure 6, there exists a branch called mask predictor M. We apply similarly to Ganea *et al.* [70] by using this two-stage fine-tuning approach. Firstly, the network is trained on base classes with lots of abundant data and then fine-tuning all predictor head C, R, and M on novel data of K shots for each class.

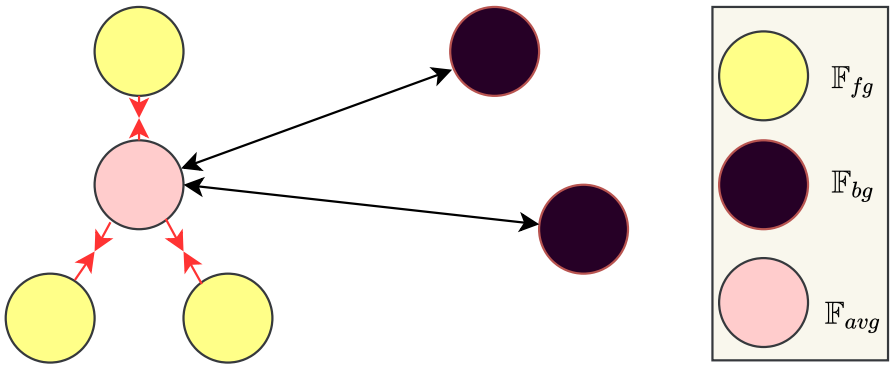


Fig. 7: Visualization of triplet loss for region proposal.

Not a simple mask predictor M that we use, we enhance the performance of the instance segmentation task by employing the two concepts of instance triplet loss and instance memory storage which are clearly described in the next section. The two improvements are inspired not only by the instance segmentation task in general but also by the camouflaged instance segmentation specifications.

3.3 Framework Improvement

One of the characteristics of camouflage instances is the camouflage texture similar to the background. This makes the precise identification of the boundary areas difficult. It is more critical in the context of few-shot learning where the concepts of a class are represented by only a few samples.

In this work, we thus propose improvements to enhance distinguishable features between background and foreground areas. In particular, we explore two approaches that focus on loss functions. The first one is the triplet loss function which was known as a strong metric to support the network in creating discrimination features between anchor and negative. The second approach is the idea of memory bank, which is used to enhance the distance between foreground and background not only for individual instances but also for each novel class. To this end, our framework is named after FS-CDIS.

To calculate the loss function, we employ the mask annotation for RoI features to collect the $\mathbb{F}_{bg}$ background and $\mathbb{F}_{fg}$ foreground features by location on each RoI. Both $\mathbb{F}_{bg}$ and $\mathbb{F}_{fg}$ for each proposal are used to calculate the respective loss functions which are presented in the following sections.

3.3.1 Instance triplet loss

With the idea of enhancing the discrimination between camouflaged instances and their backgrounds, we leverage the power of the triplet loss function [71]. Specifically, we treat the pixels of an object as positive points and the background as negative ones. Accordingly, we force the model to learn the distinguished features among the foreground and background representatives. The more distinguished among features,

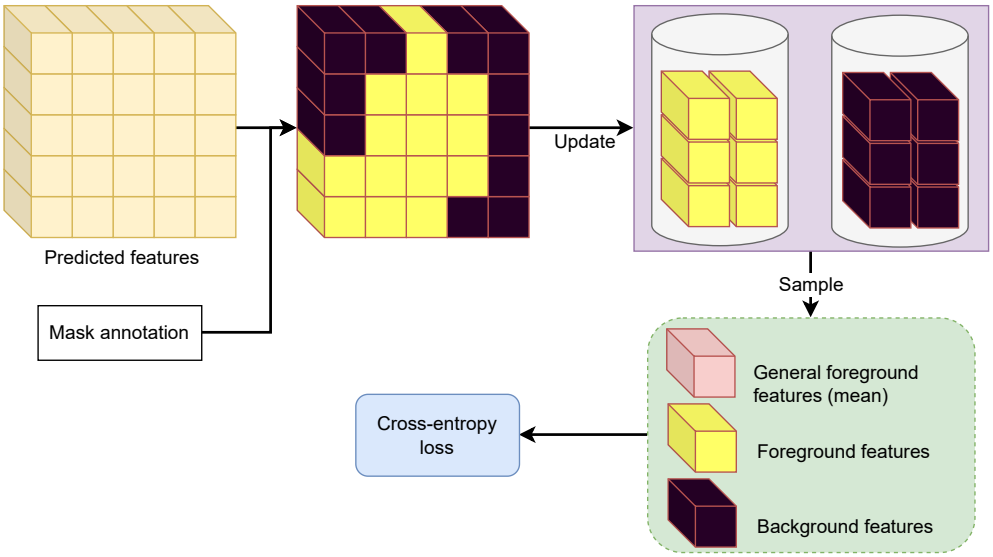


Fig. 8: Visualization of memory loss for region proposal.

the better a model can do to detect or segment camouflaged instances. In this way, we highlight the camouflaged instances so that the model is able to recognize them.

For each RoI, we consider the average foreground features $\mathbb{F}_{avg} = \frac{1}{|\mathbb{F}_{fg}|}\sum \mathbb{F}_{fg}$ as anchors with the foreground feature $\mathbb{F}_{fg}$ as positive and the background feature $\mathbb{F}_{bg}$ as negative to apply the triplet loss function [71]. In this way, the model tries to learn to minimize the distance between foreground representatives and maximize the distance between background representatives as shown in Figure 7. We use cosine similarity to calculate the distance instead of Euclidean distance. The loss function is defined as:

$$\begin{aligned} \mathcal{L}_{triplet} &= \max\{d(\mathbb{F}_{avg}, \mathbb{F}_{fg}) - d(\mathbb{F}_{avg}, \mathbb{F}_{bg}) + margin, 0\} \\ d(x, y) &= 1 - \frac{x \cdot y}{\|x\| \cdot \|y\|} \end{aligned} \tag{1}$$

, where $margin$ controls the discrimination between foreground and background features. In our experiments, we set $margin$ of 0.5.

3.3.2 Instance memory storage

The memory bank is designed to store information within a class and the class information is updated during the training. Still, the model can learn information at a global level and has high consistency for each class. On the other hand, storing and updating the features in the memory bank for each iteration during training also create more variants. By leveraging these advantages, we propose the memory bank for few-shot camouflage instance segmentation. To be specific, we use the memory bank

to contain the background and foreground features per each class and make use of features to calculate the discrimination between areas of object and no object in region proposals (shown in Figure 8).

Storing and updating: The memory bank for each class contains 2N features including N of foreground features and N of background features. While the memory bank receives new features, the module concatenates them with existing old features. In case the number of features is greater than the given N features, the memory bank releases the oldest features to maintain the number of features to N. This process updates the features in the memory bank and keeps the quantity of the stored features appropriate to the memory size (also known as the memory capacity).

Sampling: To calculate the loss value, the memory bank has to provide three elements $\mathbb{F}_{fg}$, $\mathbb{F}_{bg}$, and $\mathbb{F}_{general}$. $\mathbb{F}_{fg}$ and $\mathbb{F}_{bg}$ are all foreground and background features that module storing. The $\mathbb{F}_{general}$ is the general foreground feature, and it is created for each class by averaging the $\mathbb{F}_{fg}$.

Let $\mathbb{F}^i_{fg}$ be the i-th foreground feature and τ be a temperature hyper-parameter in [72]. In our experiments, we set τ as 1. The memory loss function for camouflaged instances is introduced as follows:

$$\mathcal{L}_{memory} = -\log \frac{\exp(\mathbb{F}_{general} \cdot \mathbb{F}^i_{fg}/\tau)}{\sum_{j=0}^{|\mathbb{F}_{bg}|} \exp(\mathbb{F}_{general} \cdot \mathbb{F}^j_{bg}/\tau) + \exp(\mathbb{F}_{general} \cdot \mathbb{F}^i_{fg}/\tau)} \tag{2}$$

To this end, the final loss of our training process, which contains an instance triplet loss and memory storage is defined as follows:

$$\mathcal{L}_{final} = \mathcal{L}_{mrcnn} + \alpha \mathcal{L}_{triplet} + \beta \mathcal{L}_{memory}. \tag{3}$$

Here, the parameter α of $\mathcal{L}_{triplet}$ and β of $\mathcal{L}_{memory}$ are used during the training process to keep the balance between the two loss functions. Details of these functions are mentioned in the following section.

4 Experiments

We first overview the metrics and the experiment settings and the implementation details in Section 4.1 and then we evaluate and discuss our improvement on the general framework, as well as ablation study for our core proposed methods in Section 4.2.

4.1 Overview

As specified in this work, we utilize the proposed CAMO-FS dataset containing images of camouflaged animals in the wild to establish the evaluation of our baseline and proposed improvement. We follow the concept procedure published in FSOD [36–38]. In the first stage of the base phase, we train our model with abundant data from 80 classes of the COCO dataset as proposed in [36]. In the second stage of the novel phase, we evaluate the performance of having $K = 1, 2, 3, 5$ shots per each novel class.

To report our results on detection and instance segmentation tasks, we use average precision (AP) and average recall (AR). To be detailed, we report AP@50 and AP@75, along with AR@10. Besides, we also report AP and AR at small, medium, and large scales of the instances to further understand the model performance. For more details,

Table 4: State-of-the-art comparison on CAMO-FS dataset among the baseline model of MTFA [70], Mask RCNN† [40], iFS-RCNN [61], and our proposed methods FS-CDIS with instance triplet loss (-ITL) and instance memory storage (-IMS). Our performance improves over the utilized baselines.

Model		Novel AP									
Method	Backbone/ Baseline	Instance Segmentation					Object Detection				
		1	2	3	5	Avg.	1	2	3	5	Avg.
MTFA [70]	COCO-80 ResNet-50	2.48	6.67	5.81	6.40	5.34	1.98	6.47	5.82	6.17	5.11
M-RCNN† [40]		4.08	6.79	6.90	8.29	6.52	2.82	5.09	5.46	6.18	4.89
iFS-RCNN [61]		4.17	6.26	5.73	6.38	5.64	3.92	6.06	5.47	6.60	5.51
MTFA [70]	COCO-80 ResNet-101	3.66	6.21	6.16	5.95	5.50	2.93	5.90	5.84	5.84	5.13
M-RCNN† [40]		4.39	7.69	7.94	10.09	7.53	3.03	5.80	6.20	7.79	5.71
iFS-RCNN [61]		4.27	6.55	6.07	7.80	6.17	3.79	6.28	6.01	8.08	6.04
Our performance (↑)											
FS-CDIS-ITL	ResNet-101 MTFA	4.46	5.57	6.41	8.48	**6.23**	4.04	7.28	7.49	9.76	**7.14**
FS-CDIS-IMS		5.46	6.95	7.36	9.61	**7.35**	4.50	6.95	7.55	10.36	**7.34**
FS-CDIS-ITL	ResNet-101 M-RCNN	5.73	7.97	8.52	9.92	**8.04**	5.08	7.56	7.85	9.67	**7.34**
FS-CDIS-IMS		5.52	7.84	8.65	9.82	**7.96**	4.92	7.39	7.96	9.52	**7.45**
FS-CDIS-ITL	ResNet-101 iFS-RCNN	5.35	6.01	7.80	6.23	**6.35**	4.71	5.66	7.10	6.06	**5.88**
FS-CDIS-IMS		2.99	6.83	6.14	9.03	**6.25**	2.74	6.39	5.94	8.44	**5.88**

M-RCNN† is Mask R-CNN [40] with sigmoid classifier.

readers can visit the homepage of the COCO dataset for detection and segmentation evaluation metrics [1].

Our MTFA [70] baseline is implemented using Detectron2 framework [73]. Our backbone is ResNet-101 [74] with Feature Pyramid Network [75]. Each experiment is set up with a single GPU GeForce RTX 2080Ti with a batch size of 2 images. The novel phase has a learning rate of 0.00125 inferred from the MTFA configuration. We set the balance parameters $\alpha = 1e^{-1}$ and $\beta = 1e^{-2}$ when we train the model with instance triplet and instance memory loss function, respectively. Please visit the publication [37] or [73] for more details on other parameters.

4.2 Results and Discussion

State-of-the-art comparison. To prove the effectiveness of our proposed methods, we conducted experiments on our proposed CAMO-FS dataset. We tested with $K = \{1, 2, 3, 5\}$ shots, respectively. Since several recent work have not published their source code [63, 64], we adopted the typical models addressing both detection and instance segmentation tasks to compare with our proposed methods. Table 4 presents the evaluation of the performance of our methods of instance triplet loss and memory storage over our baseline MTFA [70], the model of Mask R-CNN [40] with sigmoid classifier, and the state-of-the-art method iFS-RCNN [61] in the approach of few-shot instance segmentation. We reported experiments on those models and chose the common COCO-80 ResNet-101 as their base model to apply our proposed methods. The details of this decision are declared in the ablation section. In terms of instance segmentation, we improved over MTFA [70], Mask RCNN† [40], and iFS-RCNN [61] by getting average AP values of $6.23\%, 8.04\%, 6.35\%$, respectively thanks to instance triplet loss,

[1] https://cocodataset.org/#detection-eval

Table 5: Our improvement of instance triplet loss and instance memory storage on MTFA [37]. The best performance is marked in **boldface**. # denotes the Number of shots, "Memory" is Instance Memory Storage and "Triplet" is Instance Triplet Loss.

#	Method	AP	AP50	AP75	APs	APm	APl	AR1	AR10	ARs	ARm	ARl
Instance Segmentation												
1	**Baseline MTFA**	3.66	5.37	4.09	22.42	4.35	2.01	11.30	13.58	25.97	**12.96**	12.53
	MTFA + Triplet	4.46	8.21	4.60	21.33	4.13	**4.01**	12.36	15.04	23.17	9.49	16.67
	MTFA + Memory	**5.46**	**9.20**	**6.17**	**27.79**	**6.20**	**4.01**	**17.08**	**19.99**	**29.41**	11.45	**20.89**
2	**Baseline MTFA**	6.21	8.92	7.28	32.64	**7.75**	3.50	18.88	21.12	**35.82**	**15.49**	20.14
	MTFA + Triplet	5.57	9.45	6.04	25.83	3.01	5.37	15.67	17.33	26.13	7.37	17.50
	MTFA + Memory	**6.95**	**10.72**	**7.60**	**33.62**	5.73	**6.44**	**20.00**	**22.15**	34.25	13.86	**20.92**
3	**Baseline MTFA**	6.16	8.95	6.68	33.74	6.19	5.08	20.25	22.95	36.83	16.31	21.63
	MTFA + Triplet	6.41	10.67	6.72	30.39	5.17	5.30	20.69	22.98	31.90	15.69	22.53
	MTFA + Memory	**7.36**	**11.23**	**8.49**	**37.03**	**6.24**	**5.64**	**24.40**	**27.69**	**38.44**	**17.02**	**26.71**
5	**Baseline MTFA**	5.95	8.67	6.94	34.71	**6.25**	4.85	21.29	24.42	36.86	**14.51**	24.83
	MTFA + Triplet	8.48	13.43	9.80	36.66	5.75	8.04	23.83	26.66	37.03	11.62	25.91
	MTFA + Memory	**9.61**	**14.61**	**11.73**	**38.60**	5.79	**10.40**	**26.65**	**30.37**	**39.21**	12.26	**30.02**
Object Detection												
1	**Baseline MTFA**	2.93	5.86	2.20	20.95	4.18	2.03	9.25	10.84	21.74	**11.49**	8.77
	MTFA + Triplet	4.04	8.65	2.98	20.50	4.90	**4.22**	12.89	**15.53**	20.73	11.45	17.46
	MTFA + Memory	**4.50**	**9.14**	**3.45**	**22.88**	**5.61**	3.54	**13.14**	15.22	**23.14**	8.78	**16.33**
2	**Baseline MTFA**	5.90	8.87	6.83	33.04	9.74	3.10	17.26	19.25	34.04	**15.74**	19.61
	MTFA + Triplet	**7.28**	**11.22**	**8.25**	32.31	**10.72**	**6.83**	**20.52**	**22.69**	32.34	14.88	**23.52**
	MTFA + Memory	6.95	10.88	7.75	**33.93**	7.49	6.81	19.84	22.01	**34.10**	15.04	21.47
3	**Baseline MTFA**	5.84	8.98	6.29	34.56	7.78	4.31	19.13	21.83	35.80	15.93	21.09
	MTFA + Triplet	7.49	**11.51**	8.23	**38.45**	8.61	**6.38**	**24.88**	**27.52**	**38.55**	17.66	27.44
	MTFA + Memory	**7.55**	11.45	**8.50**	38.07	**9.21**	5.70	24.20	27.29	38.50	**18.10**	**27.56**
5	**Baseline MTFA**	5.84	9.13	6.04	35.44	8.17	4.22	19.67	22.96	35.94	**14.16**	22.58
	MTFA + Triplet	9.76	14.37	11.12	**40.05**	**8.82**	9.89	25.93	29.28	**40.05**	12.53	30.32
	MTFA + Memory	**10.36**	**16.27**	**11.79**	39.32	8.08	**11.36**	**26.34**	**30.30**	39.35	12.37	**30.91**

and 7.35%, 7.96%, 6.25%, respectively thanks to instance memory storage. Regarding object detection, our FS-CDIS got average amounts of 7.14%, 7.34%, 5.88%, respectively with instance triplet loss and 7.34%, 7.45%, 5.88%, respectively with instance memory storage. The detailed performance of our methods is in Table 4. Despite the limited results, we defeated the very early models on detection and instance segmentation tasks on camouflaged images.

Proposed modules evaluation. In Table 5, we also present the results of the baseline MTFA [70] with its original default configuration along with our proposed improvements. On top of the baseline MTFA [70], we establish fine-tuning configuration on this model by training all heads of classification, box regression, and mask prediction on few-shot novel data. The reported results prove the performance of the proposed instance triplet loss and instance memory storage.

In general, our approaches achieve outstanding results in comparison with the baseline. Our improvements surpass MTFA by a remarkable margin. Regarding the instance segmentation, our method improves 1.9%, 3.5%, and 2.3% in terms of AP, AP@50, and AP@75, respectively. These results manifest the efficiency of our methods in the context of few-shot camouflaged instances. Both loss functions enhance the discrimination between foreground and background features which strongly supports the model to segment pixels that belong to the camouflaged animals. Regarding the memory bank and the triplet loss function, the results of the memory loss function are higher than those of the triplet loss function by about 1%. We realize that storing representatives for each class is a crucial element in few-shot learning. This technique not

Table 6: Ablation study on the base model with 1-shot results. The best, and second best performances are marked in **red**, and *blue*, respectively. "Memory" is Instance Memory Storage and "Triplet" is Instance Triplet Loss.

Method	Base Model	Segmentation			Detection		
		AP	AP50	AP75	AP	AP50	AP75
Triplet	COCO-80 R-101	**4.46**	**8.21**	**4.60**	**4.04**	**8.65**	**2.98**
	COCO-80 R-50	3.68	*6.79*	3.81	2.85	*6.67*	1.65
	COCO-60 R-101	*3.87*	6.26	*3.90*	*3.37*	6.51	*2.69*
	COCO-60 R-50	2.56	4.25	2.79	2.28	4.13	2.26
Memory	COCO-80 R-101	**5.46**	**9.20**	**6.17**	**4.50**	**9.14**	**3.45**
	COCO-80 R-50	*3.87*	*6.81*	*3.91*	*3.40*	*6.94*	2.76
	COCO-60 R-101	2.89	4.50	3.26	2.76	4.66	*2.81*
	COCO-60 R-50	2.63	4.50	3.02	2.25	4.50	1.65

also expands the variants during training but also increases the consistency per class, thereby model can segment difficult objects better. In these ways, we also improve the corresponding results in camouflage object detection.

In Table 5, our improvements help the model segment animals in various sizes. Specifically, all three metrics including APs, APm, and APl improve in comparison with the baseline model, which demonstrates that our model well segments small, medium, and large animals. This situation also happens in the detection problem. When data is very scarce as in a 1-shot or 2-shot setting, the triplet loss function has comparative results with the memory loss function. However, in the context of 3-shot or 5-shot settings, the memory loss function demonstrates outstanding efficiency thanking to storing and updating the memory via iterations to create discriminative features on a global level. Figure 9 illustrates the qualitative comparison among the results of 5-shot settings of the baseline MTFA [70] and our proposed methods of Instance Triplet Loss and Instance Memory Storage. We chose to visualize the images with the confidence threshold of 0.5, which released a huge number of predictions with low confidence from the models. The two final rows are exemplary cases that either triplet loss or memory storage cannot well handle these camouflaged instances.

Base model ablation study. We also conduct ablation experiments on different backbone base models of the COCO settings including general and few-shot concepts. To be detailed, we report the performance of our proposed method of instance triplet loss and instance memory storage over four different backbones. The considered backbones are ResNet-50 and ResNet-101 [74]. The two base datasets are MS-COCO with 80 classes and 60 classes, respectively. Thus, it led to the combination of four different base models (i.e. COCO-80 R-101, COCO-80 R-50, COCO-60 R-101, and COCO-60 R-50). As can be seen from Table 6, the performance of applying COCO-80 R-101 base weight yields better results among others evaluated on AP, AP@50, and AP@75 in both segmentation and detection tasks. In both cases of our two proposed improvements, the ablation results demonstrate our selection of COCO-80 R-101 is the best among the tested backbones of the base phase. For the segmentation task, we achieve 4.46 and 5.46 of AP reported for triplet loss and memory storage, respectively. For the detection task, we reach 4.04 and 4.50 also of AP for the two proposals, respectively.

Table 7: Ablation study on the margin and the α ratio of the instance triplet loss in 1-shot settings. The best, and second best performances are marked in **red**, and *blue*, respectively.

AP	Segmentation					Detection				
α \ Margin	**0**	**0.25**	**0.50**	**0.75**	**1.00**	**0**	**0.25**	**0.50**	**0.75**	**1.00**
1	3.89	4.50	3.92	**5.16**	4.43	3.34	3.65	3.22	**4.22**	*3.68*
$1e^{-1}$	**4.82**	*4.74*	4.46	*4.58*	**4.57**	**4.36**	**4.27**	*4.04*	*4.16*	**3.79**
$1e^{-2}$	*4.29*	**4.74**	**4.69**	4.46	*4.39*	*4.02*	*3.97*	**4.24**	4.06	3.71

Table 8: Ablation study on the capacity of the instance memory storage in 1-shot settings. The best, and second best performances are marked in **red**, and *blue*, respectively.

Capacity	Segmentation			Detection		
	AP	**AP50**	**AP75**	**AP**	**AP50**	**AP75**
32	4.56	7.30	5.02	3.85	7.72	2.91
64	4.51	*7.67*	4.49	3.94	*8.37*	2.84
128	4.53	7.55	4.87	4.13	7.98	3.62
256	4.56	7.50	5.02	4.01	8.22	3.39
512	**4.76**	7.57	**5.37**	**4.48**	8.25	**4.44**
1024	*4.72*	**8.06**	*5.20*	*4.14*	**8.45**	*3.79*

In summary, the chosen backbone of the base weight presents a higher performance of around 1% to 2% of evaluated on common metrics as in the table. To be explained, the base from COCO-80 contains more semantic concepts in comparison with the COCO-60 base, which leads to the higher performance reported. Note that all the results in this ablation section are reported for the 1-shot setting.

Ablation on instance triplet loss component. In terms of the instance triplet loss described in Eq. 1, we establish ablation experiments to evaluate the performance of the model with different configurations of margin and α value. To this end, we set up the margin varying from 0 to 1, with a step of 0.25. For the α ratio of the loss function (Eq. 3), we check out $\alpha = \{1, 1e^{-1}, 1e^{-2}\}$. To be enhanced, the margin value indicates how distinguished foreground and background features are. Meanwhile, the α controls the effect of the instance triplet loss on the total loss of the framework. Table 7 presents the evaluation of both detection and segmentation issues in 1-shot manner. As can be inferred from the table, the effect of α decides which margin should be selected for the triplet loss. With $\alpha = 1$ meaning we keep the original ratio of the loss, the segmentation result in 1-shot setting yields the highest performance of 5.16% mAP with a 0.75 margin value. Meanwhile, the detection result gets the highest performance of 4.36% with $\alpha = 1e^{-1}$ and zero margin. This table offers a better understanding of the impact of α and the margin over the total performance.

Ablation on instance memory storage component. As for the instance memory storage, as introduced in Eq. 2, and Eq. 3, there are several parameters that need analyzing, listed as the amount of capacity in the memory storage and the β ratio

Table 9: Ablation study on the β ratio of the instance memory loss (Eq. 3) in 1-shot settings. The best, and second best performances are marked in **red**, and *blue*, respectively.

β	Segmentation			Detection		
	AP	**AP50**	**AP75**	**AP**	**AP50**	**AP75**
$1e^{-1}$	3.36	6.58	2.91	3.69	8.02	2.90
$1e^{-2}$	*4.57*	*8.02*	*4.74*	3.73	7.78	2.76
$1e^{-3}$	4.51	7.15	4.67	3.87	7.16	3.51
$1e^{-4}$	**5.12**	**8.71**	**5.54**	**4.58**	**9.23**	**3.69**
$1e^{-5}$	4.44	7.58	3.89	*4.06*	*7.99*	*3.63*

controlling the effect of the memory storage loss in the total loss. Table 8, and Table 9 present the ablation experimental results of those issues, respectively. In terms of the capacity of the memory storage, we establish experiments on a range of memory capacity of 2^i where $i = \{5, 6, 7, 8, 9, 10\}$. The reported results figure out that the performance on both segmentation and detection tasks increases with a larger capacity of memory storage. To be detailed, with a capacity of 512, the mAP metric achieves the highest value among configurations, i.e. 4.76% and 4.48% for segmentation and detection, respectively. Empirically, we select 512 to be the suitable capacity of the memory storage, not the largest. To this end, the larger capacity can confuse the model in the process of learning when retrieving information in such a large memory bank. Besides, Table 9 expresses the effectiveness of the memory loss to the total loss function. As can be inferred, $\beta = 1e^{-4}$ gives the best performance evaluated on mAP, AP50, and AP75 among all configurations.

5 Conclusion

In this work, we investigated the interesting yet challenging problem of few-shot learning for camouflaged animal detection and segmentation. We first collect a new dataset, CAMO-FS, for benchmarking purposes. We then propose a novel method to efficiently detect and segment the camouflaged animals in the images. In particular, we introduce the instance triplet loss and the instance memory storage. The extensive experiments demonstrated that our proposed method achieves state-of-the-art performance on the newly constructed dataset. We expect our work will encourage more research work in this field. In the future, we would like to extend our work with more shots for new classes. In addition, we aim to improve the computational model by taking the context into consideration.

Data Availability

The data will be made available upon request from the authors.

Fig. 9: Qualitative comparison among the selected baseline MTFA [70] and our proposed methods. The results are from 5-shot settings. "Memory" denotes Instance Memory Storage and "Triplet" denotes Instance Triplet Loss. Predicted images are visualized with a confidence threshold of 0.5, which released a huge number of predictions with low confidence from the models. The two final rows indicate exemplary cases that either triplet loss or memory storage fails to handle camouflaged instances.

Acknowledgments

This research was supported by the VNUHCM-University of Information Technology's Scientific Research Support Fund.

References

[1] Singh, S., Dhawale, C., Misra, S.: Survey of object detection methods in camouflaged image. IERI Procedia **4**, 351–357 (2013)

[2] Le, T.-N., Nguyen, T.V., Nie, Z., Tran, M.-T., Sugimoto, A.: Anabranch network for camouflaged object segmentation. CVIU **184**, 45–56 (2019)

[3] Le, T.-N., Nguyen, H.H., Yamagishi, J., Echizen, I.: Openforensics: Large-scale challenging dataset for multi-face forgery detection and segmentation in-the-wild. In: ICCV (2021)

[4] Kervrann, C., Heitz, F.: A markov random field model-based approach to unsupervised texture segmentation using local and global spatial statistics. IEEE TIP **4**(6), 856–862 (1995)

[5] Boykov, Y., Funka-Lea, G.: Graph cuts and efficient nd image segmentation. IJCV **70**(2), 109–131 (2006)

[6] Li, X., Sahbi, H.: Superpixel-based object class segmentation using conditional random fields. In: ICASSP, pp. 1101–1104 (2011)

[7] Sulimowicz, L., Ahmad, I., Aved, A.: Superpixel-enhanced pairwise conditional random field for semantic segmentation. In: ICIP, pp. 271–275 (2018)

[8] Galun, M., Sharon, E., Basri, R., Brandt, A.: Texture segmentation by multiscale aggregation of filter responses and shape elements. In: ICCV, pp. 716–723 (2003)

[9] Song, L., Geng, W.: A new camouflage texture evaluation method based on wssim and nature image features. In: International Conference on Multimedia Technology, pp. 1–4 (2010)

[10] Xue, F., Yong, C., Xu, S., Dong, H., Luo, Y., Jia, W.: Camouflage performance analysis and evaluation framework based on features fusion. Multimedia Tools and Applications **75**, 4065–4082 (2016)

[11] Pan, Y., Chen, Y., Fu, Q., Zhang, P., Xu, X.: Study on the camouflaged target detection method based on 3d convexity. Modern Applied Science **5**(4), 152 (2011)

[12] Liu, Z., Huang, K., Tan, T.: Foreground object detection using top-down information based on em framework. IEEE TIP **21**(9), 4204–4217 (2012)

[13] P. Sengottuvelan, A.W., Shanmugam, A.: Performance of decamouflaging through exploratory image analysis. In: ICETET, pp. 6–10 (2008)

[14] Yin, J., Han, Y., Hou, W., Li, J.: Detection of the mobile object with camouflage color under dynamic background based on optical flow. Procedia Engineering **15**, 2201–2205 (2011)

[15] Gallego, J., Bertolino, P.: Foreground object segmentation for moving camera sequences based on foreground-background probabilistic models and prior probability maps. In: ICIP, pp. 3312–3316 (2014)

[16] Fan, D.-P., Ji, G.-P., Sun, G., Cheng, M.-M., Shen, J., Shao, L.: Camouflaged object detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 2777–2787 (2020)

[17] Song, L., Geng, W.: A new camouflage texture evaluation method based on wssim and nature image features. In: 2010 International Conference on Multimedia Technology, pp. 1–4 (2010). IEEE

[18] Siricharoen, P., Aramvith, S., Chalidabhongse, T., Siddhichai, S.: Robust outdoor human segmentation based on color-based statistical approach and edge combination. In: The 2010 International Conference on Green Circuits and Systems, pp. 463–468 (2010). IEEE

[19] Galun, M., Sharon, E., Basri, R., Brandt, A.: Texture segmentation by multiscale aggregation of filter responses and shape elements. In: ICCV, vol. 3, p. 716 (2003)

[20] Kavitha, C., Rao, B.P., Govardhan, A.: An efficient content based image retrieval using color and texture of image sub-blocks. International Journal of Engineering Science and Technology (IJEST) **3**(2), 1060–1068 (2011)

[21] Xue, F., Yong, C., Xu, S., Dong, H., Luo, Y., Jia, W.: Camouflage performance analysis and evaluation framework based on features fusion. Multimedia Tools and Applications **75**(7), 4065–4082 (2016)

[22] Hou, J.Y.Y.H.W., Li, J.: Detection of the mobile object with camouflage color under dynamic background based on optical flow. Procedia Engineering **15**, 2201–2205 (2011)

[23] Skurowski, P., Abdulameer, H., Baszczyk, J., Depta, T., Kornacki, A., Kozie, P.: Animal camouflage analysis: Chameleon database. Unpublished Manuscript (2018)

[24] Zhai, Q., Li, X., Yang, F., Chen, C., Cheng, H., Fan, D.-P.: Mutual graph learning for camouflaged object detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 12997–13007 (2021)

[25] Li, A., Zhang, J., Lv, Y., Liu, B., Zhang, T., Dai, Y.: Uncertainty-aware joint salient object and camouflaged object detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 10071–10081 (2021)

[26] Mei, H., Ji, G.-P., Wei, Z., Yang, X., Wei, X., Fan, D.-P.: Camouflaged object segmentation with distraction mining. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 8772–8781 (2021)

[27] Yan, J., Le, T.-N., Nguyen, K.-D., Tran, M.-T., Do, T.-T., Nguyen, T.V.: Mirrornet: Bio-inspired camouflaged object segmentation. IEEE Access **9**, 43290–43300 (2021)

[28] Zhu, J., Zhang, X., Zhang, S., Liu, J.: Inferring camouflage objects by texture-aware interactive guidance network. In: AAAI (2021)

[29] Lv, Y., Zhang, J., Dai, Y., Li, A., Liu, B., Barnes, N., Fan, D.-P.: Simultaneously localize, segment and rank the camouflaged objects. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 11591–11601 (2021)

[30] Price, N., Green, S., Troscianko, J., Tregenza, T., Stevens, M.: Background matching and disruptive coloration as habitat-specific strategies for camouflage. Scientific reports **9**(1), 1–10 (2019)

[31] Le, T.-N., Nguyen, V., Le, C., Nguyen, T.-C., Tran, M.-T., Nguyen, T.V.: Camoufinder: Finding camouflaged instances in images. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, pp. 16071–16074 (2021)

[32] Le, T.-N., Cao, Y., Nguyen, T.-C., Le, M.-Q., Nguyen, K.-D., Do, T.-T., Tran, M.-T., Nguyen, T.V.: Camouflaged instance segmentation in-the-wild: Dataset, method, and benchmark suite. IEEE Transactions on Image Processing **31**, 287–300 (2022)

[33] Pia Bideau, E.L.-M.: It's moving! a probabilistic model for causal motion segmentation in moving camera videos. In: ECCV (2016)

[34] Lamdouar, H., Yang, C., Xie, W., Zisserman, A.: Betrayed by motion: Camouflaged object discovery via motion segmentation. In: ACCV (2020)

[35] Fan, Q., Zhuo, W., Tang, C.-K., Tai, Y.-W.: Few-shot object detection with attention-rpn and multi-relation detector. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2020)

[36] Kang, B., Liu, Z., Wang, X., Yu, F., Feng, J., Darrell, T.: Few-shot object detection via feature reweighting. In: ICCV (2019)

[37] Wang, X., Huang, T.E., Darrell, T., Gonzalez, J.E., Yu, F.: Frustratingly simple few-shot object detection. In: ICML (2020)

[38] Yan, X., Chen, Z., Xu, A., Wang, X., Liang, X., Lin, L.: Meta r-cnn: Towards general solver for instance-level low-shot learning. In: ICCV (2019)

[39] Redmon, J., Farhadi, A.: Yolo9000: better, faster, stronger. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 7263–7271 (2017)

[40] He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask r-cnn. In: ICCV, pp. 2980–2988 (2017)

[41] Li, B., Yang, B., Liu, C., Liu, F., Ji, R., Ye, Q.: Beyond max-margin: Class margin equilibrium for few-shot object detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2021)

[42] Hu, H., Bai, S., Li, A., Cui, J., Wang, L.: Dense relation distillation with context-aware aggregation for few-shot object detection. In: Proceedings of (CVPR) (2021)

[43] Xiao, Y., Marlet, R.: Few-shot object detection and viewpoint estimation for objects in the wild. In: European Conference on Computer Vision, pp. 192–210 (2020). Springer

[44] Han, G., He, Y., Huang, S., Ma, J., Chang, S.-F.: Query adaptive few-shot object detection with heterogeneous graph convolutional networks. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 3263–3272 (2021)

[45] Han, G., Huang, S., Ma, J., He, Y., Chang, S.-F.: Meta faster r-cnn: Towards accurate few-shot object detection with attentive feature alignment. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 36, pp. 780–789 (2022)

[46] Li, A., Li, Z.: Transformation invariant few-shot object detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2021)

[47] Wu, J., Liu, S., Huang, D., Wang, Y.: Multi-scale positive sample refinement for few-shot object detection. In: ECCV (2020)

[48] Zhang, G., Luo, Z., Cui, K., Lu, S.: Meta-detr: Image-level few-shot object detection with inter-class correlation exploitation. arXiv preprint arXiv:2103.11731 (2021)

[49] Han, G., Ma, J., Huang, S., Chen, L., Chang, S.-F.: Few-shot object detection with fully cross-transformer. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 5321–5330 (2022)

[50] Sun, B., Li, B., Cai, S., Yuan, Y., Zhang, C.: Fsce: Few-shot object detection via contrastive proposal encoding. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 7352–7362 (2021)

[51] Zhang, S., Wang, L., Murray, N., Koniusz, P.: Kernelized few-shot object detection with efficient integral aggregation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 19207–19216 (2022)

[52] Qiao, L., Zhao, Y., Li, Z., Qiu, X., Wu, J., Zhang, C.: Defrcn: Decoupled faster r-cnn for few-shot object detection. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 8681–8690 (2021)

[53] Zhang, W., Wang, Y.-X.: Hallucination improves few-shot object detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 13008–13017 (2021)

[54] Zhu, C., Chen, F., Ahmed, U., Shen, Z., Savvides, M.: Semantic relation reasoning for shot-stable few-shot object detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2021)

[55] Khandelwal, S., Goyal, R., Sigal, L.: Unit: Unified knowledge transfer for any-shot object detection and segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2021)

[56] Liu, W., Zhang, C., Lin, G., Liu, F.: Crnet: Cross-reference networks for few-shot segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2020)

[57] Dong, N., Xing, E.P.: Few-shot semantic segmentation with prototype learning. In: BMVC, vol. 3 (2018)

[58] Wang, K., Liew, J.H., Zou, Y., Zhou, D., Feng, J.: Panet: Few-shot image semantic segmentation with prototype alignment. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) (2019)

[59] Saha, O., Cheng, Z., Maji, S.: Ganorcon: Are generative models useful for few-shot segmentation? In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 9991–10000 (2022)

[60] Tian, Z., Lai, X., Jiang, L., Liu, S., Shu, M., Zhao, H., Jia, J.: Generalized few-shot semantic segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 11563–11572 (2022)

[61] Nguyen, K., Todorovic, S.: ifs-rcnn: An incremental few-shot instance segmenter. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 7010–7019 (2022)

[62] Gao, B.-B., Chen, X., Huang, Z., Nie, C., Liu, J., Lai, J., Jiang, G., Wang, X., Wang, C.: Decoupling classifier for boosting few-shot object detection and instance segmentation. In: NeurIPS 2022 (2022)

[63] Han, Y., Zhang, J., Xue, Z., Xu, C., Shen, X., Wang, Y., Wang, C., Liu, Y., Li, X.: Reference twice: A simple and unified baseline for few-shot instance segmentation. arXiv preprint arXiv:2301.01156 (2023)

[64] Wang, H., Liu, J., Liu, Y., Maji, S., Sonke, J.-J., Gavves, E.: Dynamic transformer for few-shot instance segmentation. In: Proceedings of the 30th ACM International Conference on Multimedia, pp. 2969–2977 (2022)

[65] Snell, J., Swersky, K., Zemel, R.: Prototypical networks for few-shot learning. Advances in neural information processing systems **30** (2017)

[66] Vinyals, O., Blundell, C., Lillicrap, T., Wierstra, D., et al.: Matching networks for one shot learning. Advances in neural information processing systems **29** (2016)

[67] Gidaris, S., Komodakis, N.: Dynamic few-shot visual learning without forgetting. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4367–4375 (2018)

[68] Fan, Z., Yu, J.-G., Liang, Z., Ou, J., Gao, C., Xia, G.-S., Li, Y.: Fgn: Fully guided network for few-shot instance segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 9172–9181 (2020)

[69] Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. In: NeurIPS, pp. 91–99 (2015)

[70] Ganea, D.A., Boom, B., Poppe, R.: Incremental few-shot instance segmentation. In: Proceedings of (CVPR), pp. 1185–1194 (2021)

[71] Balntas, V., Riba, E., Ponsa, D., Mikolajczyk, K.: Learning local feature descriptors with triplets and shallow convolutional neural networks. In: Bmvc, vol. 1, p. 3 (2016)

[72] Wu, Z., Xiong, Y., Yu, S.X., Lin, D.: Unsupervised feature learning via non-parametric instance discrimination. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3733–3742 (2018)

[73] Wu, Y., Kirillov, A., Massa, F., Lo, W.-Y., Girshick, R.: Detectron2. https://github.com/facebookresearch/detectron2 (2019)

[74] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016)

[75] Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: CVPR (2017)