Few Shot Object Detection Using Fine-tuning

Jean-Bernard Uwineza
University of California, Riverside
Riverside, CA 92501
buwineza@ee.ucr.edu

Chetan Reddy Mudireddy
University of California, Riverside
Riverside, CA 92501
cmudi001@ucr.edu

Om Shankar Ohdar
University of California, Riverside
Riverside, CA 92501
oohda001@ucr.edu

Sayak Nag
University of California,Riverside
Riverside, CA 92501
snag005@ucr.edu

Vikarn Bhakri
University of California, Riverside
Riverside, CA 92501
vbhak001@ucr.edu

Abstract

Despite the recent overwhelming success of deep convolutional neural networks in object detection, they still require large amounts of annotated examples per category to perform well. Few-shot object detection aims to detect rare object categories that often have only a few examples, and might not be present in training examples. Most approaches have focused on meta-learning techniques such as feature re-weighting, or feature re-generation. This report explores a technique of finetuning only the last layer of the detection network. It is shown that this gives overwhelmingly better results despite being relatively simple. The network was trained on base classes of PASCAL VOC, and COCO datasets containing general classes. We additionally show that the model is able to generalize by evaluating it on novel classes from the KITTI dataset.

1 Introduction

Few-shot learning has received significant interest in the past few years in computer vision, but mainly for the tasks of classification and rarely for object detection. The task of object detection is inherently more challenging since the detector not only has to perform recognition of the different kinds of objects present, it also has to localize them. This is already a challenging task that relies heavily on the availability of massive amounts of labeled training data. When a new data-point is obtained belonging to a novel category, adapting the model becomes a very difficult task especially when the new category contains a few samples. Recently, meta learning techniques have been proposed for adapting deep models to novel categories. However, they are not easily extendable to the task of object detection. Matching Networks [1] and Prototypical Networks [2], building prototypes of objects is much more difficult than building prototype of the categories. Another approach that is being explored by researchers is to provide ways to fine-tune the detection layers of deep models to adapt to the new categories [3].

In this project we aim to explore the state of the art fine-tuning approach proposed in [3]. Additionally, we introduce a different similarity metric and show that it can improve detection in some instances. We experiment on benchmark datasets such as COCO [4] and PASCAL [5] and also extend these

Submitted as final report of Spring 2020 EE 260 Project.

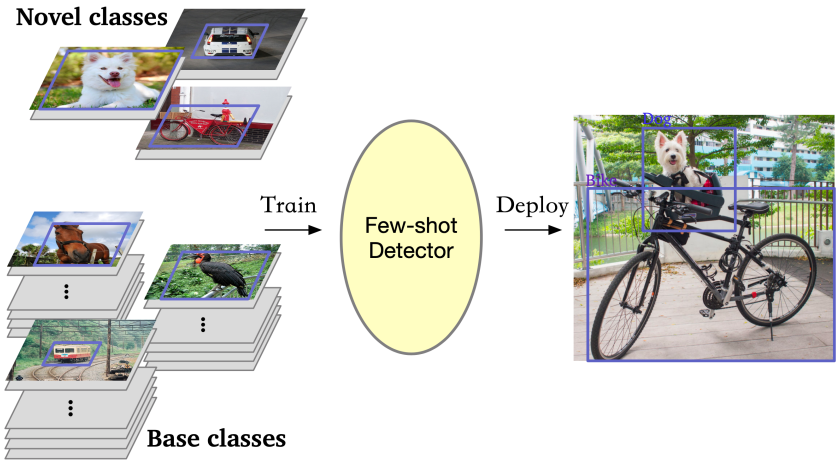


Figure 1: A typical few-shot object detection process. Base classes contain abundant training examples per object category, while novel class only have a few examples per category. The few-shot detector must generalize well enough to detect novel objects from only a few examples.

approaches towards object detection with the KITTI dataset [6]. Additionally, we examine how many shots are necessary to reach comparable accuracy relative to conventional detection approaches.

2 Existing Approaches

2.1 General Detectors

There are many approaches to the general object detection problem. The most prominent and successful ones use deep convolutional neural networks (Deep CNNs). A series of related detectors in [7–9] began with R-CNN, a network which uses pre-trained CNNs to classify region proposals [7]. The proposals are generated using a method of selective search introduced in [10]. Fast R-CNN [8] added a region-of-interest (RoI) pooling layer to extract regional features directly from convolutional maps. Faster R-CNN [9] introduced an entire region proposal network (RPN) to speed region proposal.

In the pursuit of yet more detection speed and simplicity, there is another group of detectors which forgoes region proposal. The YOLO series of detectors begins with [11], a detector which uses a single convolutional network to directly perform class and bounding box predictions. Incremental improvements are introduced in [12, 13]. Since these methods do not require a separate classifier for each region, they are simpler and faster than proposal-based methods.

Since few-shot detection in not mainly concerned with speed, it borrows many techniques from proposal-based approaches. Specifically, RoI layers and RPN networks will be used in the approach explored in this paper (see Section 3.2).

2.2 Few-shot Detectors

Early work on few-shot learning used Bayesian inference as an attempt to generalize knowledge learned from a pre-trained model [14]. Currently, meta-learning approaches are the most popular. Particularly, a subclass called metric learning performs detection using a similarity metric. Matching networks in [1] attempts to learn the similarity between the target image and a small set of labeled images. This idea is extended by Prototypical Networks [2] by using a linear classifier instead of nearest-neighbor for each class. All the above are for the classification task.

For detection, [15] defines a setting in which two kinds of training data are available: *base classes* and *novel classes*. This is shown in Figure 1. Base classes have abundant labeled samples, while novel classes have only a limited number of samples. The job of the detector is to use knowledge of base classes to learn how to detect novel objects in a test setting that includes both base and novel classes. This is motivated partly by the fact that large-scale object detection datasets (such as PASCAL VOC

and MSCOCO) only contain a limited number of object categories. However, these datasets can be used in training as base classes to support the introduction of novel classes.

In meta-learning, a *meta-leaner* is used to acquire class-level meta-knowledge and help the model generalize to novel classes through various techniques such as feature re-weighting [15] or class-specific feature generation [16]. The meta-learner trains on a small set of support images with full annotations of target objects, and is often jointly trained with the detector using episodic training, in which each episode comprises of N supporting objects a set of few query images [1]. The training is also divided into two stages; in the first stage the model is only trained on base classes, and in the second stage the model is fine-tuned on a support set of a few examples of the novel classes and a small subset of the base classes.

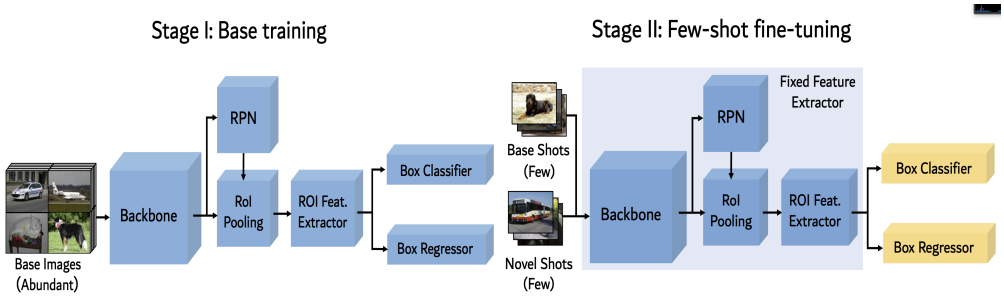
The approach explored herein adopts this two-stage training regime, but it gets rid of memory inefficiency of episodic training. [3]. Instead, the fine-tuning is only done on the last layers of the network with a normal batch training scheme (See Section 3.2).

3 Few-shot Finetuning

3.1 Problem Definition

Suppose there is a set of base classes C_b with *sufficiently many* training examples and a set C_n of novel classes with only K examples per class (in this paper $K \leq 10$ always). The few-shot datasets used herein are balanced, meaning that for each novel class there are K annotated examples. When K examples are used in finetuning, we call it a k-shot detection. Denote the set of training images as $\mathcal{X}$ and their corresponding labels as $\mathcal{Y}$. A detection dataset $\mathcal{D} = \{(x, y),\ x \in \mathcal{X},\ y \in \mathcal{Y}\}$, where x is the input image and $y = \{(c_i, \mathbf{1}_i)\}_{i=1}^{N}$ denotes the categories $c \in C_b \cup C_n$ and bounding box coordinates $\mathbf{1}$ of the N instances of the object in the image x.

In [1] and [2], as is usually used in few-shot learning, the authors use N-way K-shot regime for evaluation, where N is the number of training examples, and K is the number of classes in the training set. In this paper, the detector model is evaluated on a test set that includes base and novel classes to optimize the detection accuracy as measured by mean average precision (mAP) of both novel and base classes.



(a) Stage I (b) Stage II

Figure 2: Stage I & II of the finetuning process. Figure courtesy of Wang et al.

3.2 Two-stage Finetuning

In a two-stage process, we adopt the simple finetuning approach of [3]. This approach builds on Faster R-CNN [9], a popular two-stage object detector model. The approach is pictured in Figure 7. Figure 2a shows that the Faster R-CNN model has three main components, namely, the backbone, the Region Proposal Network (RPN), and the two-layer fully-connected network as a Region of Interest (RoI) feature extractor. The backbone used in this paper is the ResNet architecture [17], but it could be any other architecture such as the VGG16. The ResNet was chosen because it is relatively easier to optimize, hence potentially easier to adapt to novel classes. In addition to Faster R-CNN model, henceforth denoted $\mathcal{F}$, the approach also uses a box classifier $\mathcal{C}$ for classifying objects into different classes and a box regressor $\mathcal{R}$ to predict the bounding boxes.

The main intuitive assumption is that $\mathcal{F}$ is able to transfer the feature function $\mathcal{F}(x)$ learned from base classes to novel classes. In other words, $\mathcal{F}$ is class-agnostic. This leads to the main contribution of [3]: a suggestion that when introducing novel classes, it matters little to retrain $\mathcal{F}$. Instead, only the the box classifier $\mathcal{C}$ and regressor $\mathcal{R}$ need to be finetuned on the examples of novel classes since the features of base classes learned with $\mathcal{F}$ will easily transfer to novel classes. In this way, feature representation learning and box prediction learning are separated into two stages, as shown in Figure 7.

Stage I: Base Training. The feature extractor and the box predictor are trained on a dataset containing base classes, C_b. The training loss function [9] is

$$L = L_{\mathcal{F}} + L_{\mathcal{C}} + L_{\mathcal{R}}. \tag{1}$$

The $L_{\mathcal{F}}$ loss is obtained from the output of the RPN network, hence, it helps distinguishing background from foreground. The $L_{\mathcal{C}}$ loss is a cross-entropy loss of the box classifier, and $L_{\mathcal{R}}$ is the robust loss function (smoothed L_1) of the box regressor. More details on L can be found in [9] and [8].

Stage II: Finetuning. In this stage, a small training set containing K examples per class, with both base and novel classes. Using a smaller learning rate[1] and the same loss function as in (1), the box classifier and regressor are initialized with random weights for the novel classes and are finetuned while the feature extractor $\mathcal{F}$ is fixed and unmodified.

3.3 Euclidean Distance-based Similarity

The weight matrix $W \in \mathbb{R}^{d\times c}$ of the box classifier $\mathcal{C}$ can be expressed as $W = [w_1, w_2, \ldots, w_c]$ where the row-vector $w_c \in \mathbb{R}^d$ is the weight vector of each class. Recent works (such as [1, 18–20]) have proposed box classifiers based on cosine similarity. The output of the classifier is a scaled similarity score matrix of the input features and the weight vectors of the classes. Specifically

$$s_{ij} = \frac{\alpha \mathcal{F}(x)_i^\top w_j}{\|\mathcal{F}(x)_i\| \|w_j\|}, \tag{2}$$

where α is the scaling factor[2]. Cosine similarity is attractive because it offers feature normalization which reduces the variance between classes and improves detection accuracy for novel classes [3].

We now propose a different similarity score, in which the Euclidean distance between the input features and class weight vectors is passed through the Softmax function. Specifically

$$s_{ij} = \frac{e^{-\|\mathcal{F}(x)_i - w_j\|_2^2}}{\sum_j e^{-\|\mathcal{F}(x)_i - w_j\|_2^2}}. \tag{3}$$

The Euclidean distance is one of the simplest distance metrics, and it has been shown in [2] to outperform cosine similarity scores. We extend this metric and find that this similarity score provides a stronger kind of feature normalization which should help the model adapt well when introducing novel classes.

4 Experimental Results

4.1 Experimental Setup

The network architecture has been provided in section 3.2. All models were trained using SGD with a minibatch size of 16, the momentum of 0.9, and a weight decay of 0.0001. A learning rate of 0.02 is used during base training and 0.001 during few-shot fine-tuning. For base training, the model was trained for 20 epochs in total with early stopping for COCO and Pascal-VOC and then fined-tuned for the novel categories for 10 epochs. For KITTI, the PASCAL-VOC base feature detector was fixed and only the last layers we fine-tuned for 20 epochs.

[1] The finetuning learning rate is reduced by at least 20 times compared to base training learning rate. In our experiments, it was reduced from $2 \cdot 10^{-2}$ to $1 \cdot 10^{-3}$.

[2] In [3], a fixed value of $\alpha = 20$ was used.

For the PASCAL VOC dataset, its 2007 and 2012 versions were used for training. Twenty of its classes were randomly divided into 15 base classes and 5 novel classes, where for each of the novel classes 1, 5 and 10 images were sampled from the validation set, corresponding to their specific class [3]. The PASCAL VOC 2007 test set is used for evaluation. For COCO, the 60 categories were used as base classes while the remaining 20 classes are used as novel classes with K = 1, 5 and 10 (number of shots). For the KITTI dataset, we took 50 images and separated them into two classes of pedestrian and cars. For KITTI the feature extractor is not trained on any base classes but instead, we use the PASCAL VOC feature extractor and then fine-tune the detectors on 20 of the KITTI images and then test the model on the remaining 30. The final accuracy is measured by average precision (AP) of the novel classes, as well as, the base classes (for PASCAL and COCO) and as the evaluation metric, we used AP50 where the matching threshold is 0.5 [3].

4.2 Baseline

As discussed in section 3.3 we used a euclidean distance over softmax as the similarity index as compared to the cosine based similarity of Wang et. al. [3]. Therefore, we compare our results with that of [3]... Our approach is compared for 1, 5 and 10 shots on the PASCAL VOC and COCO dataset. Our implementation showed an improvement in 2 out of three K-shot settings. Further, we also tested our approach on the real-world dataset of traffic images, the KITTI dataset for 1, 5 and 10 shots.

5 Results

The mean average precision (mAP) is used to evaluate the results of the detectors. It gives a number between 0 and 100 (the higher the better) and is different compared to the accuracy metric in classification. Each of the bounding boxes in a output image has a score associated with it(likelihood of the box containing an object). For each of these predictions the area under the Precision-recall Curve is obtained which i known as Average Precision (AP). By averaging the AP values over all the classes in the image gives us mAP. A detection is a true positive if it has an 'intersection over union' (IoU or overlap) with the ground-truth box greater than some threshold. In our case we used a threshold of 0.5 and hence we also called it mAP50. The IoU is defined as follows,

$$IoU = \frac{ground\ truth \cap prediction}{ground\ truth \cup prediction} \tag{4}$$

We provide the mAP50 results on all the datasets used in table 1. Since KITTI was not used by Wang et.al. [3] we do not add their results and only report ours.

	COCO		**PASCAL**		**KITTI**
	Ours	Wang et. al.	Ours	Wang et. al.	Ours
1-shot	37.1	39.8	36.3	39.8	22.5
5-shot	**56.3**	55.7	**47.3**	45.2	31.7
10-shot	54.9	56	**48.1**	42.2	29.8

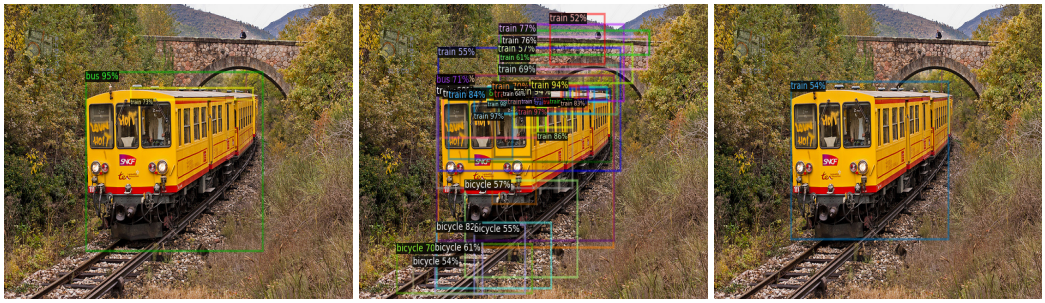
Table 1: mAP50 results on PASCAL, COCO and KITTI test sets. Instances where our approach is better is shown in bold.

In general increasing the number of shots (i.e. data samples for the novel classes) helps in improving the detection results. This can be observed from figures 3, 4, 5. In case of COCO we observed that the detector was producing a lot of bounding boxes with greater than 50% prediction scores.

In general using the detector of [3] with our similarity index gives comparable results to their cosine distance based similarity. As observed in table 1 our similarity index tends to give better performance for the cases when more than one shot of novel (class) training data was used. More results can be found in Appendix A.

We provide the mAP50 of the COCO test set in Table 1 for the three K-shot settings. We were able to achieve better performance on the 5-shot and 10-shot cases.

We provide the mAP50 of the KITTI test-set in Table 1 for the three K-shot settings. Since KITTI was not used in [3] in their experiments, we are not reporting the results the detector would have achieved with a cosine similarity based classifier and we just report the mAP50 of our approach.

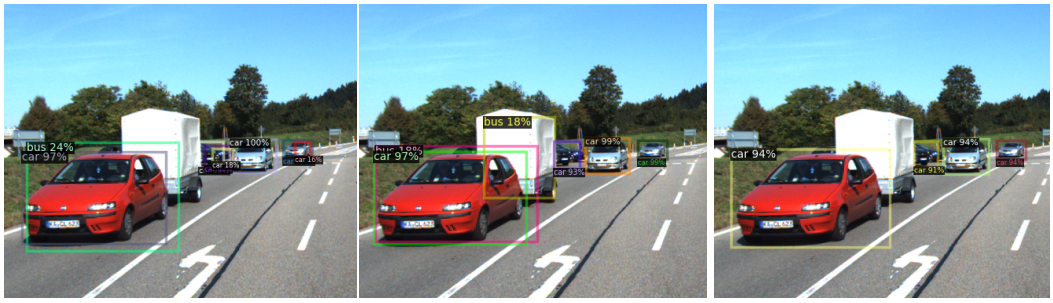


(a) 1-shot (b) 5-shot (c) 10-shot

Figure 3: Detection Results on COCO dataset.

(a) 1-shot (b) 5-shot (c) 10-shot

Figure 4: Detection Results on PASCAL dataset.



(a) 1-shot (b) 5-shot (c) 10-shot

Figure 5: Detection Results on KITTI dataset.

6 Conclusion

In this project we analysed the state of the art few shot object detection technique of Wang et. al. [3]. Instead of using the cosine distance based similarity of [3] for obtaining the object wise classification score, we used Euclidean distance over softmax to obtain the classification score. We also showed the application of this few-shot object detection approach for detecting cars and pedestrians from traffic images (KITTI). As observed from our results in table 1 our similarity index not only gives comparable performance to that of [3] but also outperforms it many cases. Our experiments show that fine tuning based few-shot object detection techniques can also be used on traffic images where obtaining manually annotated data is challenging. Extension of these techniques for 3D object detection can also be explored where using Lidar based depth maps can help provide more context for the model to better segment the objects in an image.

References

[1] O. Vinyals, C. Blundell, T. P. Lillicrap, K. Kavukcuoglu, and D. Wierstra, "Matching networks for one shot learning," *CoRR*, vol. abs/1606.04080, 2016. [Online]. Available: http://arxiv.org/abs/1606.04080

[2] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," in *Advances in neural information processing systems*, 2017, pp. 4077–4087.

[3] X. Wang, T. E. Huang, T. Darrell, J. E. Gonzalez, and F. Yu, "Frustratingly Simple Few-Shot Object Detection," *arXiv preprint arXiv:2003.06957*, 2020. [Online]. Available: http://arxiv.org/abs/2003.06957

[4] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: common objects in context," *CoRR*, vol. abs/1405.0312, 2014. [Online]. Available: http://arxiv.org/abs/1405.0312

[5] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal Visual Object Classes (VOC) Challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, Jun. 2010.

[6] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets Robotics: The KITTI Dataset," *International Journal of Robotics Research (IJRR)*, 2013.

[7] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[8] R. Girshick, "Fast R-CNN," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[9] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[10] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders, "Selective search for object recognition," *International journal of computer vision*, vol. 104, no. 2, pp. 154–171, 2013.

[11] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[12] J. Redmon and A. Farhadi, "Yolo9000: better, faster, stronger," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7263–7271.

[13] ——, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[14] L. Fei-Fei, R. Fergus, and P. Perona, "One-shot learning of object categories," *IEEE transactions on pattern analysis and machine intelligence*, vol. 28, no. 4, pp. 594–611, 2006.

[15] B. Kang, Z. Liu, X. Wang, F. Yu, J. Feng, and T. Darrell, "Few-shot object detection via feature reweighting," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 8420–8429.

[16] Y.-X. Wang, D. Ramanan, and M. Hebert, "Meta-learning to detect rare objects," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 9925–9934.

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[18] W.-Y. Chen, Y.-C. Liu, Z. Kira, Y.-C. F. Wang, and J.-B. Huang, "A closer look at few-shot classification," *arXiv preprint arXiv:1904.04232*, 2019.

[19] H. Qi, M. Brown, and D. G. Lowe, "Low-shot learning with imprinted weights," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 5822–5830.

[20] S. Gidaris and N. Komodakis, "Dynamic few-shot visual learning without forgetting," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4367–4375.

A More Results

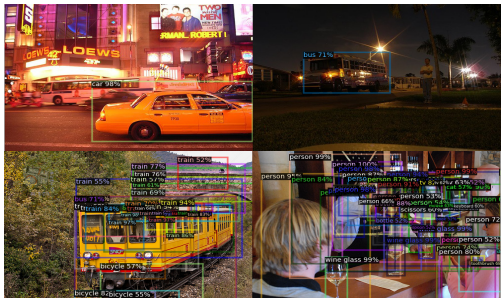
(a)

(b)

(c)

Figure 6: Our Results on the PASCAL Dataset. (a) 1-shot (b) 5-shot (c) 10-shot

(a)



(b)

(c)

Figure 7: Our Results on the COCO Dataset. (a) 1-shot (b) 5-shot (c) 10-shot

(a)

(b)

(c)

Figure 8: Our Results on the KITTI Dataset. (a) 1-shot (b) 5-shot (c) 10-shot