Hallucination Improves Few-Shot Object Detection Supplementary Material

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

A. Additional Implementation Details

The hallucinator used in the main results is a two-layer MLP. The first linear layer consists of three blocks: class prototype block, seed example block, and noise block, which take as inputs a class prototype, a seed example, and a noise vector, respectively. We adopt a similar strategy as in [3] to initialize the hallucinator. We initialize the class prototype block and the seed example block by an identity mapping plus small random noise. We initialize the noise block by small random noise. We initialize the second linear layer by an identity mapping plus small random noise. The initialization noise is sampled from a normal distribution with a zero mean and a standard deviation of 0.02.

B. Significance of Improvement

In Table A, we compare the scale of improvements on COCO with the standard deviation computed from three groups of CoRPNs [4], each with a different set of hyper-parameters. We observe a notable (at least one standard deviation) and stable improvement with hallucination, indicating that hallucination positively impacts performance.

C. Additional Analysis on Different Types of Hallucinators

In the main paper, we have investigated two types of hallucinators – conservative and aggressive hallucinators. Here we provide a more in-depth analysis of the two hallucinators and *focus on the impact of different design choices*. First, the feature space for hallucination is one of the primary factors in designing a hallucinator. While hallucination is restricted to the classifier's feature space for the conservative hallucinator, the aggressive hallucinator generates examples in the feature space before the box head and thus affects the components in the box head as well. Because of the difference in the hallucination space, the hallucinator architectures also vary from each other. The conservative hallucinator consists of two fully-connected layers, denoted as 'Halluc after box head (2-fc)' in Table B; the aggressive hallucinator consists of three convolutional layers, denoted as 'Halluc before box head (3-conv)'.

Figure A. 1-shot detection results of TFA [2] (**rows 1 and 3**) and 'TFA + Halluc' (**rows 2 and 4**) on PASCAL VOC under base/novel split 1. Novel classes are {bird, bus, cow, motorbike, sofa}. As discussed in the main paper, 'TFA + Halluc' *significantly improves the true positive detection* while *eliminating many false positives*, compared with the TFA baseline.

Second, the hallucinator architecture also impacts its training procedure. Since the box head is shared by both the box classifier and the box regressor, fine-tuning the box head using the EM-style training procedure is not suitable for the aggressive hallucinator. Therefore, we jointly train the aggressive hallucinator with other model components. Specifically, in joint training, the hallucinator is *trained together* with the classifier instead of being trained from an already-trained classifier. We further explore different types of training losses. For example, 'Prototype cosine dist. (joint)' in Table B trains the hallucinator using the prototypical network loss with cosine distance [1]. The seed examples and the hallucinated examples are used as the support set to construct batch-wise prototypes; we compute the

	Method	1-shot AP	1-shot AP50	1-shot AP75	2-shot AP	2-shot AP50	2-shot AP75	3-shot AP	3-shot AP50	3-shot AP75
Ours	CoRPNs w/ cos + Halluc	**4.38**	**7.48**	**4.88**	**5.58**	**9.86**	**5.85**	**7.20**	**13.27**	**7.44**
Baseline	CoRPNs w/ cos [4], reported	4.13	7.20	4.37	5.41	9.58	5.62	7.06	13.19	7.24
Mean	3 × CoRPNs w/ cos [4]	4.21	7.15	4.59	5.23	9.27	5.50	7.06	13.06	7.09
Standard deviation	3 × CoRPNs w/ cos [4]	0.20	0.13	0.36	0.12	0.25	0.11	0.13	0.13	0.26

Table A. Comparison of the performance gain of hallucination with the standard deviation computed from three groups of CoRPNs on PASCAL VOC under three base/novel splits. **Bottom two rows:** the mean and standard deviation obtained by three CoRPNs models, each with a different hyper-parameter setting. **Top two rows:** results from CoRPNs and 'CoRPNs + Halluc', using *the same set of hyper-parameters*. 'CoRPNs + Halluc' consistently outperforms CoRPNs by at least one standard deviation. Since 'CoRPNs + Halluc' and CoRPNs *share the same RPN outputs*, the results imply that hallucination significantly improves CoRPNs.

	Loss		
Architecture	Cross-entropy (EM)	Cross-entropy (joint)	Prototype cosine dist. (joint)
Halluc before box head (3-conv)	–	33.7	**40.8**
Halluc after box head (2-fc)	**45.1**	**41.9**	33.8

Table B. Impact of different design choices of hallucinators on PASCAL VOC 1-shot AP50 results under train/novel split 1. Results in **bold** outperform the TFA baseline (39.8).

Hallucinator	Novel Set 1	Novel Set 2	Novel Set 3
Random noise	42.9	24.1	37.7
Random interpolation	46.7	22.0	32.2
Parametric (Ours)	**47.0**	**26.3**	**40.4**

Table C. 1-shot AP50 comparison of three hallucinators with CoRPNs on PASCAL VOC under three base/novel splits. Our parametric hallucinator outperforms the two non-parametric hallucinator baselines under all three base/novel splits.

loss based on held-out query boxes. The other two training loss options are 'Cross-entropy (EM)' which is the standard EM-style training procedure described in the main paper, and 'Cross-entropy (joint)' which jointly trains the hallucinator and the classifier using a cross-entropy loss.

Table B together with Table 5 in the main paper summarizes the impact of the different design choices. Our main observations are as follows. (1) Theses design choices of the hallucinator, including the architecture, training procedure, and training loss, *all make a difference* on the detection performance. (2) The training procedure and loss are *highly coupled with* the hallucinator architecture. For example, jointly training the hallucinator and the classifier is beneficial to the aggressive hallucinator. (3) Learning a *unified* hallucinator across different datasets is an interesting direction, which we leave as future work.

D. Comparison with Non-parametric Hallucinators

We compare our parametric hallucinator with two non-parametric hallucinators. The *random noise hallucinator* generates examples by adding small random noise to seed examples. The *random interpolation hallucinator* generates examples by interpolating between random seed example pairs. For a fair comparison, all three hallucinators produce the same number of examples per batch. As shown in Table C, our parametric hallucinator *consistently outperforms* non-parametric hallucinator baselines.

Figure B. 1-shot detection results of TFA [2] (**rows 1 and 3**) and 'TFA + Halluc' (**rows 2 and 4**) on COCO. Novel classes shown here are {tv, bus, horse}. As discussed in the main paper, 'TFA + Halluc' *significantly improves the true positive detection* while *eliminating many false positives*, compared with the TFA baseline.

E. Visualizations

We provide some final detection visualizations on PASCAL VOC (Figure A) and COCO (Figure B). Consistent with the observations in the main paper, hallucination significantly improves the true positive detection while eliminating many false positives.

References

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

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

[3] Yu-Xiong Wang, Ross Girshick, Martial Hebert, and Bharath Hariharan. Low-shot learning from imaginary data. In *CVPR*, 2018. 1

[4] Weilin Zhang, Yu-Xiong Wang, and David A. Forsyth. Cooperating RPN's improve few-shot object detection. *arXiv preprint arXiv:2011.10142*, 2020. 1, 2