Supplementary Material of Few-shot Open-set Recognition by Transformation Consistency

Minki Jeong Seokeon Choi Changick Kim
Korea Advanced Institute of Science and Technology, Daejeon, Republic of Korea
{rhm033, seokeon, changick}@kaist.ac.kr

1. Training Details

1.1. Training Setups

We organize optimizer details to train the feature extractor F and the transformer $\boldsymbol{T}$ in Table 1. The first pre-training is done up to 500 epochs using the whole base dataset, like regular supervised classification training. For instance, we trained the backbone network with a simple 64 classes classifier for the miniImageNet dataset. We set a single epoch to 100 episodes for training our modules. The validation is performed after every training epochs. We combine the OSR methods that use the prediction probability to detect unknowns [2, 7], which observed that unseen class samples are likely to have a low prediction probability. We use random cropping, random scaling, random color jittering, and random horizontal flipping to augment base data. During the evaluation, we only use central cropping to supports and queries.

We employed a temperature value in the distance function for stable training. It is formulated as follows:

$$dist(\boldsymbol{a}, \boldsymbol{b}) = ||(\boldsymbol{a} - \boldsymbol{b})||_2^2/\tau, \tag{1}$$

where τ is a temperature value, set to 64.

1.2. Modification on the FSL and OSR methods

In this subsection, we describe the details of our adjustments of the FSL and OSR methods for the FSOSR comparison. The FSL methods [9, 12] create a logit vector with negatives of distances between a query feature and prototypes. Then they apply the softmax function on the logit to calculate classification probabilities. Based on the OSR methods, we use the negative of the prediction probability to detect unknowns for FSL methods.

OpenMax [1] fits a Weibull model using logit vectors for each known class. However, this is not directly applicable to FSL problems since a class set formation differs by episode. To this end, we fit Weibull models of relative classes. For each episode, we assign relative labels to the episodes, from one to five. The logit vectors are used to fit Weibull models of the assigned classes, and mean activation vectors for unknown class sample detection. NN [4] measures the distance between the query feature and the nearest class feature, and the distance between that feature and the second closest class feature. Then, the ratio of the distances determines unknowns. We use distances from prototypes to calculate the ratio.

2. Experimental Details

2.1. Detailed Results

We provide the detailed experimental results on miniImageNet and tieredImageNet with the confidence intervals in Table 2 and Table 3, respectively. SnaTCHer excels in both classification and detection.

2.2. Backbone architecture

We selected the ResNet-12 based architecture as a feature extractor for a fair comparison, following previous FSL methods [5, 12]. We further explored the relationship between the backbone architecture and network performances by exchanging the backbone to the ResNet-18-based architecture. Table 4 shows the ablation study result. The ResNet-18 backbone shows slightly worse classification performance than the ResNet-12 backbone. Thus, we selected the ResNet-12-based architecture as the feature extractor.

2.3. Transformation analysis

We illustrate prototypes before and after the transformation to confirm the role of the transformation in Fig. 1. We utilize SnaTCHer-F for the visualization. The difference of the known-replaced set (*i.e.*, between squares and crosses) is smaller than that of the unknown-replaced set (*i.e.*, between circles and crosses). The known query feature and the unknown query feature are close in the projected figure. Therefore it is hard to classify knowns and unknowns in the naïve approach. However, the difference after the transformation is notably increased after the relationship-based feature transformation. Since the transformation considers the

	F initialization	F	$\boldsymbol{T}$
Optimizer type	SGD	SGD	SGD
Learning rate	0.1	1×10^{-4}	1×10^{-3}
Momentum	0.9	0.9	0.9
Weight decay	5×10^{-4}	5×10^{-4}	5×10^{-4}
Learning rate scheduler	MultiStep	Step	Step
Scheduler steps	[350, 450, 440, 460, 480]	40	40
Scheduler gamma	0.1	0.5	0.5
Total epochs	500	200	200

Table 1. Optimizer details

	miniImageNet 5-way			
	1-shot		5-shot	
Model	Acc	AUROC	Acc	AUROC
ProtoNet [9]	64.01 ± 0.88	51.81 ± 0.93	80.09 ± 0.58	60.39 ± 0.92
FEAT [12]	67.02 ± 0.85	57.01 ± 0.84	82.02 ± 0.53	63.18 ± 0.78
NN [4]	63.82 ± 0.85	56.96 ± 0.75	80.12 ± 0.57	63.43 ± 0.76
OpenMax [1]	63.69 ± 0.84	62.64 ± 0.80	80.56 ± 0.58	62.27 ± 0.71
PEELER* [6]	58.31 ± 0.58	61.66 ± 0.62	75.08 ± 0.72	69.85 ± 0.70
PEELER [6]	65.86 ± 0.85	60.57 ± 0.83	80.61 ± 0.59	67.35 ± 0.80
SnaTCHer-F	67.02 ± 0.85	68.27 ± 0.96	82.02 ± 0.53	$\mathbf{77.42 \pm 0.73}$
SnaTCHer-T	66.60 ± 0.80	$\mathbf{70.17 \pm 0.88}$	81.77 ± 0.53	76.66 ± 0.78
SnaTCHer-L	$\mathbf{67.60 \pm 0.83}$	69.40 ± 0.92	$\mathbf{82.36 \pm 0.58}$	76.15 ± 0.83

Table 2. Average closed-set classification accuracies (%) and average unknown detection AUROCs (%) on miniImageNet over 600 episodes. PEELER* is quoted from the paper, which has a ResNet-18 backbone.

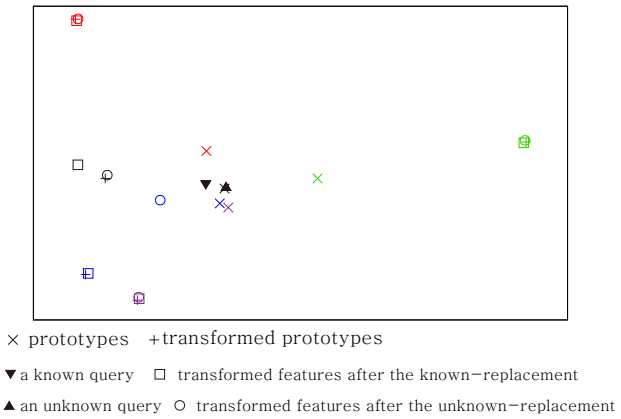

Figure 1. PCA projected prototypes before and after the modification. The colors indicate classes of features. Squares indicate the transformed features after exchanging the known query and its corresponding prototype, and circles indicate that of the unknown query.

relationships of all prototypes, the unknown sample fails to reconstruct the set properly even though it can reconstruct its predicted class prototype correctly (see the blue class example).

2.4. Distance function

We utilized the Euclidean distance function for a fair comparison with previous methods for logit calculations, and used it for the unseen sample detection for consistency with logits. We further investigated the influence of the different distance function choices on the unseen sample detection. Table 5 shows the performance changes when we use cosine distance for SnaTCHer-L. As pointed in various FSL studies, the distance function impacts overall performances.

2.5. Cross-domain FSOSR

We provide the detailed result of Table 3 in the manuscript in Table 6.

Furthermore, we extended the cross-domain FSOSR evaluation on Meta-Dataset [10]. Meta-Dataset consists of train/validation/test splits of various datasets, including ImageNet [8] and CUB [11]. All models are trained with 5-shot episodes from the ImageNet train split for evaluations. We prepare three evaluation scenarios for cross-domain FSOSR. The first case samples knowns from the test split of ImageNet, and collects unknown instances from the train splits of CUB presented in Meta-Dataset. We denote it as an ImageNet-CUB case. Similarly, we define a CUB-ImageNet case and a CUB-CUB case to assess the generalization capabilities of the FSOSR methods. Table 7

	tieredImageNet 5-way			
	1-shot		5-shot	
Model	Acc	AUROC	Acc	AUROC
ProtoNet [9]	68.26 ± 0.96	60.73 ± 0.80	83.40 ± 0.65	64.96 ± 0.83
FEAT [12]	70.52 ± 0.96	63.54 ± 0.76	84.74 ± 0.69	70.74 ± 0.75
NN [4]	67.73 ± 0.96	62.70 ± 0.72	83.43 ± 0.66	69.77 ± 0.75
OpenMax [1]	68.28 ± 0.95	60.13 ± 0.74	83.48 ± 0.66	65.51 ± 0.83
PEELER [6]	69.51 ± 0.92	65.20 ± 0.76	84.10 ± 0.66	73.27 ± 0.71
SnaTCHer-F	70.52 ± 0.96	74.28 ± 0.80	84.74 ± 0.69	82.02 ± 0.64
SnaTCHer-T	70.45 ± 0.95	74.84 ± 0.79	84.42 ± 0.68	$\mathbf{82.03 \pm 0.66}$
SnaTCHer-L	$\mathbf{70.85 \pm 0.99}$	$\mathbf{74.95 \pm 0.83}$	$\mathbf{85.23 \pm 0.64}$	80.81 ± 0.68

Table 3. Average closed-set classification accuracies (%) and average unknown detection AUROCs (%) on tieredImageNet over 600 episodes.

	1-shot		5-shot	
Architecture	Acc(%)	AUROC(%)	Acc(%)	AUROC(%)
ResNet-12				
PEELER [6]	65.86 ± 0.85	60.57 ± 0.83	80.61 ± 0.59	67.35 ± 0.80
SnaTCHer-F	67.02 ± 0.85	68.27 ± 0.96	82.02 ± 0.53	$\mathbf{77.42 \pm 0.73}$
SnaTCHer-T	66.60 ± 0.80	$\mathbf{70.17 \pm 0.88}$	81.77 ± 0.53	76.66 ± 0.78
SnaTCHer-L	$\mathbf{67.60 \pm 0.83}$	69.40 ± 0.92	$\mathbf{82.36 \pm 0.58}$	76.15 ± 0.83
ResNet-18				
PEELER* [6]	58.31 ± 0.58	61.66 ± 0.62	75.08 ± 0.72	69.85 ± 0.70
PEELER [6]	64.03 ± 0.84	59.92 ± 0.85	77.60 ± 0.62	62.89 ± 0.83
SnaTCHer-F	66.83 ± 0.81	67.33 ± 0.78	$\mathbf{81.74 \pm 0.55}$	$\mathbf{76.71 \pm 0.75}$
SnaTCHer-T	$\mathbf{66.86 \pm 0.83}$	68.13 ± 0.99	80.56 ± 0.62	76.66 ± 0.77
SnaTCHer-L	66.38 ± 0.86	$\mathbf{69.19 \pm 0.96}$	81.09 ± 0.57	76.10 ± 0.81

Table 4. Backbone architecture comparison on miniImageNet 5-way tasks. PEELER* is quoted from the paper.

Function	5-way 1-shot	5-way 5-shot
Euclidean	$\mathbf{69.40 \pm 0.92}$	$\mathbf{76.15 \pm 0.83}$
Cosine	68.60 ± 0.94	75.45 ± 0.86

Table 5. AUROC (%) differences on the distance function. We used 600 miniImageNet episodes for the comparison.

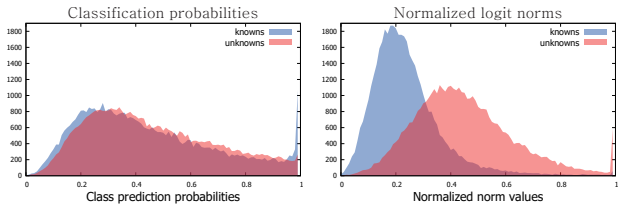

Figure 2. The histograms of the normalized logit norm values and classification probabilities of PEELER on the 600 5-way 1-shot CUB-ImageNet episodes. The norm values are normalized to the maximum norm value of the episode.

shows the result. When the known and unknown domains are different, it is easier to detect unknowns. Therefore AUROC values are large, even over 90% in the ImageNet-CUB 5-shot case. Overall, our methods show higher AUROC values with comparable or better classification performance than PEELER for all cases. On top of that, using the prediction probability with the softmax operation shows poor performance in the cross-domain cases. We illustrated the histograms of normalized logit norm values and classification probabilities of PEELER in Fig. 2. The smaller logit norms indicate that the query sample is closer to the prototypes. Since knowns are closer than the unknowns, the logit norms are well separated. However, the softmax operation uses relative distance information to calculate class prediction probabilities. This sacrifices the distribution information of samples, which makes it hard to distinguish known samples from unknown samples. Therefore, using absolute distances is better than using relative functions such as the softmax function.

3. Layer-Task Normalization details

In this section, we provide deep analysis of our transformation method (LTN).

3.1. Weight generator details

LTN uses a weight generator, which creates a balance parameter between the layer normalized and instance normalized features. We describe the details of the generator in Fig. 3. The max-pooling layer makes the generator symmetric.

Model	tieredImageNet-CUB				CUB-tieredImageNet				CUB-CUB			
	5-way 1-shot		5-way 5-shot		5-way 1-shot		5-way 5-shot		5-way 1-shot		5-way 5-shot	
	Acc	AUROC	Acc	AUROC	Acc	AUROC	Acc	AUROC	Acc	AUROC	Acc	AUROC
PEELER [6]	69.51 ± 0.92	67.59 ± 0.88	84.10 ± 0.66	76.10 ± 0.87	58.81 ± 0.88	57.58 ± 0.69	77.66 ± 0.73	64.38 ± 0.73	59.42 ± 0.91	58.63 ± 0.68	78.42 ± 0.70	66.04 ± 0.70
SnaTCHer-F	70.52 ± 0.96	83.22 ± 0.83	84.74 ± 0.69	90.12 ± 0.59	57.81 ± 0.92	63.47 ± 0.78	77.33 ± 0.70	69.64 ± 0.68	57.98 ± 0.87	64.55 ± 0.75	77.05 ± 0.69	71.05 ± 0.73
SnaTCHer-T	70.45 ± 0.95	$\mathbf{84.95 \pm 0.70}$	84.42 ± 0.68	$\mathbf{91.83 \pm 0.47}$	57.84 ± 0.93	$\mathbf{64.52 \pm 0.76}$	77.77 ± 0.70	$\mathbf{70.63 \pm 0.70}$	57.82 ± 0.89	$\mathbf{65.10 \pm 0.76}$	77.66 ± 0.70	$\mathbf{72.04 \pm 0.66}$
SnaTCHer-L	$\mathbf{70.85 \pm 0.99}$	83.67 ± 0.82	$\mathbf{85.23 \pm 0.64}$	90.23 ± 0.58	$\mathbf{60.21 \pm 0.90}$	63.55 ± 0.73	$\mathbf{79.27 \pm 0.66}$	69.42 ± 0.66	$\mathbf{59.69 \pm 0.92}$	64.75 ± 0.74	$\mathbf{78.68 \pm 0.67}$	70.67 ± 0.66

Table 6. Detailed result of Table 3 in the manuscript.

Model	ImageNet-CUB				CUB-ImageNet				CUB-CUB			
	5-way 1-shot		5-way 5-shot		5-way 1-shot		5-way 5-shot		5-way 1-shot		5-way 5-shot	
	Acc	AUROC	Acc	AUROC	Acc	AUROC	Acc	AUROC	Acc	AUROC	Acc	AUROC
PEELER [6]	50.51 ± 0.91	58.63 ± 1.21	73.38 ± 0.69	73.78 ± 1.01	62.28 ± 0.92	47.97 ± 1.23	82.33 ± 0.65	59.40 ± 1.47	62.62 ± 0.94	57.26 ± 0.81	82.44 ± 0.67	65.61 ± 0.88
SnaTCHer-F	$\mathbf{59.86 \pm 0.96}$	92.82 ± 0.48	77.14 ± 0.71	97.93 ± 0.14	65.79 ± 0.97	98.49 ± 0.17	84.52 ± 0.70	99.32 ± 0.10	66.92 ± 0.93	67.05 ± 0.81	84.73 ± 0.69	74.33 ± 0.68
SnaTCHer-T	59.05 ± 0.94	$\mathbf{95.47 \pm 0.49}$	$\mathbf{77.26 \pm 0.69}$	$\mathbf{98.94 \pm 0.16}$	66.02 ± 0.95	98.66 ± 0.19	86.54 ± 0.63	99.37 ± 0.10	65.48 ± 0.97	68.27 ± 0.85	85.80 ± 0.62	75.91 ± 0.68
SnaTCHer-L	58.86 ± 0.90	93.35 ± 0.70	76.86 ± 0.71	98.26 ± 0.24	$\mathbf{68.11 \pm 0.90}$	$\mathbf{98.96 \pm 0.15}$	$\mathbf{87.36 \pm 0.57}$	$\mathbf{99.48 \pm 0.10}$	$\mathbf{67.90 \pm 0.97}$	$\mathbf{69.88 \pm 0.78}$	$\mathbf{87.30 \pm 0.61}$	$\mathbf{77.49 \pm 0.64}$

Table 7. Cross-domain FSOSR comparison results on Meta-Dataset.

3.2. Fusing the normalization methods

In this subsection, we assess the effect of fusing instance normalization and layer normalization. At first, we illustrate a histogram of the generated weight values in Fig. 4 to analyze the importance of the normalization methods. High weight values indicate that the contribution of LN is higher than IN. This observation corresponds to the performance difference between IN and LN in Table 1 in the manuscript, which shows that LN is better than IN.

We further investigate the contribution of IN and LN by fixing the weight value to zero and one, respectively. Table 8 shows the comparison result. Here SnaTCHer-L-LN, SnaTCHer-L-IN, and SnaTCHer-L share the same parameters except for the balance weight. SnaTCHer-L-LN set the weight to one to assess layer normalization. On the contrary, SnaTCHer-L-IN fixes the weight value to zero to use instance normalization only. According to the table, using IN or LN alone is worse than use them both. This result validates our approach to use different normalization methods altogether for better performances. Moreover, the classification performance gap is larger in the 1-shot case. Since supports in 1-shot scenarios are not enough to represent class characteristics than supports of 5-shot cases, the combination with IN compensates exceptional support instances of a class with large intra-class variance.

3.3. Classification way dependence

We study performance differences by changing the way of episodes. Figure 5 shows the comparison result. During the comparison, we use the network trained with 1-shot 5-way episodes. The number of unknown classes is fixed to five, and the number of supports per class is set to one for all cases. The classification accuracy drops as the way increases because the task becomes much more challenging. The detection AUROC also decreases since the inter-class variance of the known classes increases. The larger intra-class variance indicates a larger known feature space, making it hard to distinguish unknown samples. Nevertheless, our method still outperforms the previous state-of-the-art FSOSR method, displaying the broad applicability of SnaTCHer.

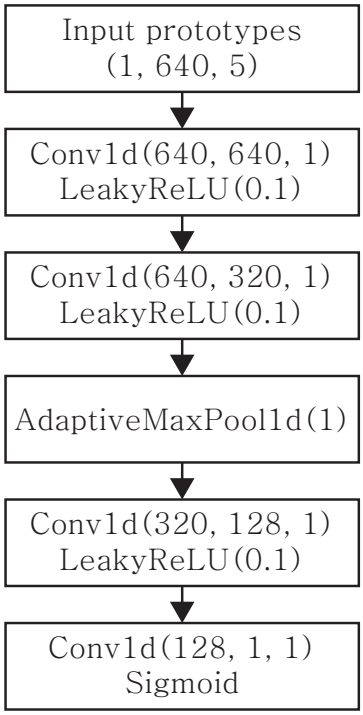

Figure 3. The weight generator structure details with a 5-way task. The layers are described in PyTorch function styles.

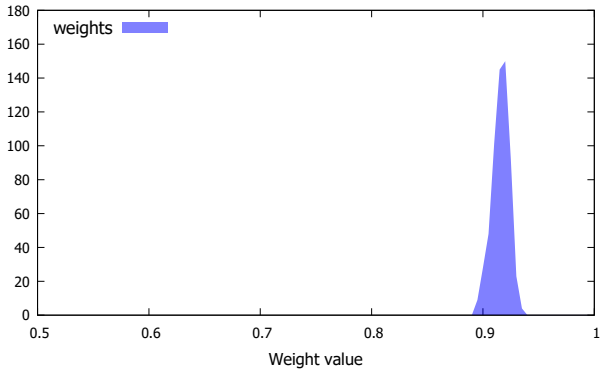

Figure 4. The histogram of the weight values over 600 5-way 5-shot tasks on tieredImageNet.

	1-shot		5-shot	
Architecture	Acc(%)	AUROC(%)	Acc(%)	AUROC(%)
SnaTCHer-L-IN	47.58 ± 1.03	66.06 ± 0.79	53.15 ± 1.18	71.01 ± 0.67
SnaTCHer-L-LN	66.88 ± 0.83	69.30 ± 0.93	82.27 ± 0.52	76.09 ± 0.84
SnaTCHer-L	$\mathbf{67.60 \pm 0.83}$	$\mathbf{69.40 \pm 0.92}$	$\mathbf{82.36 \pm 0.58}$	$\mathbf{76.15 \pm 0.83}$

Table 8. Normalization analysis on miniImageNet 5-way tasks. SnaTCHer-L-IN and SnaTCHer-L-LN use IN and LN only, respectively. All parameters are trained with SnaTCHer-L.

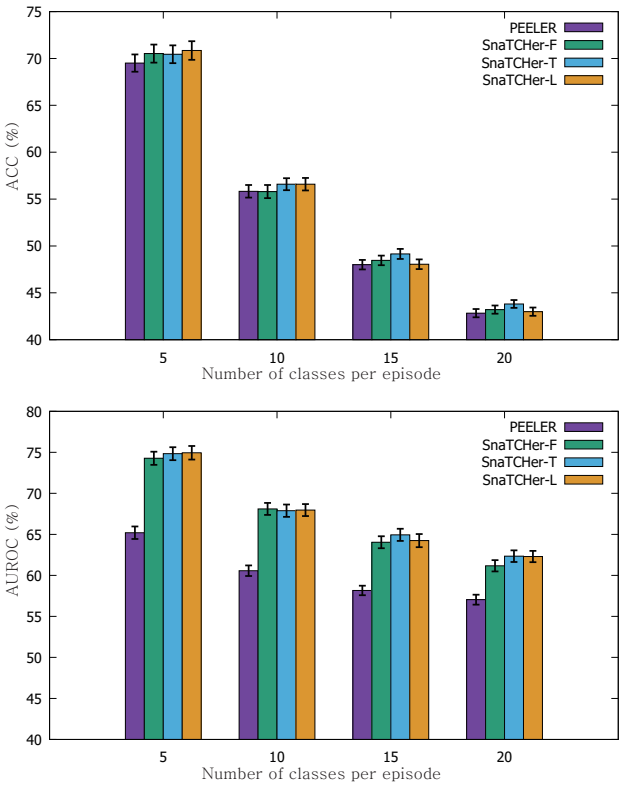

Figure 5. The performances changes over different way configurations. The network is trained on 1-shot 5-way tasks. We set different number of classes (5, 10, 15, 20) to compare performances.

4. Threshold analysis

The unseen detection performance varies by the threshold value even if we use the same model with the same parameters. This nature interferes with a fair comparison of unseen sample detection methods. Recent studies have proposed various evaluation methods to reduce the effect of evaluation hyperparameter settings during comparisons.

In the manuscript, we employed AUROC to compare performances. AUROC measures false positive rate and true positive rate changes by varying the threshold values, free from the threshold decision. Therefore, it is widely used in recent methods to compare unseen sample detection performances [7, 6, 3]. In line with previous studies, we utilized AUROC to compare detection performances.

Architecture	Accuracy (%)	F1-score	AUROC (%)
SnaTCHer-F	55.44	0.6904	82.02
SnaTCHer-T	54.42	0.6865	**82.03**
SnaTCHer-L	**70.82**	**0.7376**	80.81

Table 9. Unseen sample detection comparison on 600 tieredImageNet 5-way 5-shot episodes.

Nevertheless, determining the threshold value for the unseen detection is important to use the models in real-world problems. Since the threshold should adapt to different episodes, we propose to use the difference between the average of prototypes and its farthest prototype for FSOSR. The comparison result is reported in Table 9. As evident from the table, the threshold decision strategy should be dependent on the transformation function for better performance. SnaTCHer-F and SnaTCHer-T show poor performances though they have similar AUROC values with the others. This observation emphasizes the need to use parameter-free measurements for performance comparisons. Furthermore, this result shows that the threshold decision method needs more attention of the society.

References

[1] Abhijit Bendale and Terrance E Boult. Towards open set deep networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1563–1572, 2016. 1, 2, 3

[2] Akshay Raj Dhamija, Manuel Günther, and Terrance Boult. Reducing network agnostophobia. In *Advances in Neural Information Processing Systems*, pages 9157–9168, 2018. 1

[3] Chuanxing Geng, Sheng-jun Huang, and Songcan Chen. Recent advances in open set recognition: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020. 5

[4] Pedro R Mendes Júnior, Roberto M De Souza, Rafael de O Werneck, Bernardo V Stein, Daniel V Pazinato, Waldir R de Almeida, Otávio AB Penatti, Ricardo da S Torres, and Anderson Rocha. Nearest neighbors distance ratio open-set classifier. *Machine Learning*, 106(3):359–386, 2017. 1, 2, 3

[5] Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, and Stefano Soatto. Meta-learning with differentiable convex optimization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10657–10665, 2019. 1

[6] Bo Liu, Hao Kang, Haoxiang Li, Gang Hua, and Nuno Vasconcelos. Few-shot open-set recognition using meta-learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8798–8807, 2020. 2, 3, 4, 5

[7] Lawrence Neal, Matthew Olson, Xiaoli Fern, Weng-Keen Wong, and Fuxin Li. Open set learning with counterfactual images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 613–628, 2018. 1, 5

[8] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015. 2

[9] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in neural information processing systems*, pages 4077–4087, 2017. 1, 2, 3

[10] Eleni Triantafillou, Tyler Zhu, Vincent Dumoulin, Pascal Lamblin, Utku Evci, Kelvin Xu, Ross Goroshin, Carles Gelada, Kevin Swersky, Pierre-Antoine Manzagol, and Hugo Larochelle. Meta-dataset: A dataset of datasets for learning to learn from few examples. In *International Conference on Learning Representations*, 2020. 2

[11] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011. 2

[12] Han-Jia Ye, Hexiang Hu, De-Chuan Zhan, and Fei Sha. Few-shot learning via embedding adaptation with set-to-set functions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8808–8817, 2020. 1, 2, 3