Domain Adaptive Few-Shot Open-Set Learning
Supplementary Material

Debabrata Pal[1], Deeptej More[2], Sai Bhargav[1], Dipesh Tamboli[3], Vaneet Aggarwal[3], Biplab Banerjee[1]
[1]Indian Institute of Technology, Bombay, [2]Manipal Institute of Technology, India,[3]Purdue University
`debabrata.pal@iitb.ac.in, deeptejrane16@gmail.com, sai.bhargav@iitb.ac.in, dtamboli@purdue.edu, vaneet@purdue.edu, getbiplab@gmail.com`

In the Supplementary Material, we provide additional contents in, (Sec: 1) Brief summary of notations, (Sec: 2) Comparison of computation complexity, (Sec: 3) Algorithms, (Sec: 4) DAFOS-NET parameter space, (Sec: 5) AUROC, (Sec: 6) Evaluation of DAFOS-NET in DA-FSL setting, (Sec: 7) Noise variance impact, (Sec: 8) Analysis of Prototype diversification loss., (Sec: 9) DA-FSOS experimental protocol and remaining comparisons on DomainNet and Office-Home, (Sec: 10) Anti open close mode collapse loss.

1. Notation table and associated descriptions

Notations	Descriptions
Source Domain ($\mathcal{S}$)	
$\mathcal{C}_s, \mathcal{D}_s$	$\mathcal{S}$ classes for training and associated training set
$\mathcal{K}_\mathcal{S}, \mathcal{U}_\mathcal{S}$	known and pseudo-unknown classes in an episode of $\mathcal{S}$
$m_\mathcal{S}$	Support samples per known class in $\mathcal{S}$
$\mathbb{S}_\mathcal{S}$	$\mathcal{S}$ domain support samples of known classes, $\mathbb{S}_\mathcal{S}(\mathcal{K}_\mathcal{S})$
$\mathbb{S}_{f\mathcal{S}}, \mathbb{S}_{l\mathcal{S}}, \mathbb{S}_{h\mathcal{S}}$	Support, pseudo-known and pseudo-unknown features of $\mathcal{S}$
$\mathcal{P}'_\mathcal{S}$ / $\mathcal{P}_\mathcal{S}$	$\mathcal{S}$ domain known class prototype without / with adaptation
$\mathbb{Q}_\mathcal{S}, \mathbb{Q}_{f\mathcal{S}}$	$\mathcal{S}$ queries of known and outliers, $\mathbb{Q}_\mathcal{S}(\mathcal{K}_\mathcal{S} \cup \mathcal{U}_\mathcal{S})$ and its features
Target Domain ($\mathcal{T}$)	
$\mathcal{C}_d, \mathcal{C}_t$	$\mathcal{T}$ classes for training and testing
$\mathcal{C}'_t, \mathcal{C}_t - \mathcal{C}'_t$	Known and unknown classes for testing
$\mathcal{D}_d, \mathcal{D}_t, \mathcal{D}'_t$	Few-shot training set, testing set, testing set with known classes
$\mathcal{K}_\mathcal{T}, \mathcal{U}_\mathcal{T}$	known and pseudo-unknown classes in an episode of $\mathcal{T}$
$m_\mathcal{T}$	Support samples per known class in $\mathcal{T}$
$\mathbb{S}_\mathcal{T}$	$\mathcal{T}$ domain support samples of known classes, $\mathbb{S}_\mathcal{T}(\mathcal{K}_\mathcal{T})$
$\mathbb{S}_{f\mathcal{T}}, \mathbb{S}_{l\mathcal{T}}, \mathbb{S}_{h\mathcal{T}}$	Support, pseudo-known and pseudo-unknown features of $\mathcal{T}$
$\mathcal{P}'_\mathcal{T}$ / $\mathcal{P}_\mathcal{T}$	$\mathcal{T}$ domain known class prototype without / with adaptation
$\mathbb{Q}_\mathcal{T}, \mathbb{Q}_{f\mathcal{T}}$	$\mathcal{T}$ queries of known and outliers, $\mathbb{Q}_\mathcal{T}(\mathcal{K}_\mathcal{T} \cup \mathcal{U}_\mathcal{T})$ and its features
Additional input data and parameters	
(x_i^s, y_i^s), (x_i^t, y_i^t)	Input image and class label for $\mathcal{S}$ and $\mathcal{T}$ domains
$\mathcal{K}$, m	m training samples selected per $\mathcal{K}$-known class, ($\mathcal{K}$-way m-shot)
q_{Pos}, q_{Neg}	Positive queries and negative queries to optimize $\mathcal{L}_{PD}$
$\mathcal{Q}_{dist}$	Euclidean distance of each query from respective prototype
Network components and labels	
f_φ	Feature extractor
$G_{\mathcal{L}\theta}, D_{\mathcal{L}\phi}$	Pseudo-known sample generator and discriminator
$G_{\mathcal{H}\theta}, D_{\mathcal{H}\phi}$	Pseudo-unknown sample generator and discriminator
$\mathcal{O}_\zeta, \mathcal{O}_\eta$	Outlier detector, Domain predictor
t_i, c_i	known / outlier label, source / target domain label
Parameters and Hyper-parameters	
$\mathcal{N}(0, \sigma)$	Isotropic Gaussian noise with mean=0, standard deviation σ
$\sigma \in \{\sigma L, \sigma H\}$	σL: Low standard deviation to generate pseudo-known samples
	σH: High standard deviation to generate pseudo-unknown samples
$\gamma_\mathcal{S}, \gamma_\mathcal{T}$	Scaling parameters of batch-norm for source and target domain
$\beta_\mathcal{S}, \beta_\mathcal{T}$	Translation parameters of batch-norm for source and target domain
z_l, z_h	Noise vectors for generating pseudo-known, and unknown samples
s_l, s_h	GAN generated pseudo-known, pseudo-unknown samples
α	Margin parameter in $\mathcal{L}_{PD}$
$\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5$	Weight factors of multiple loss components in optimizing $\mathcal{L}_C$
Loss functions	
$\mathcal{L}_l, \mathcal{L}_h$	Pseudo known and unknown generating cGAN training losses.
$\mathcal{L}_{FE}$	Feature extractor loss
$\mathcal{L}_C$,$\mathcal{L}_{PD}$	Known-class compactness loss, Prototype diversification loss
$\mathcal{L}_{Align}$	Global cross-domain prototype alignment loss
$\mathcal{L}_{OUT}, \mathcal{L}_{DC}$	Outlier detection loss, Domain prediction loss
$\mathcal{L}_{AOCMC}$	Anti open close mode collapse loss

2. Computational complexity

We evaluate different models on NVIDIA 3080 Ti having 24GB graphics memory and compare their computational cost and space complexity in Table 1. MetaoptNet demands heavy computational resources as it has a 0.7 GFLOP value and 18.1 million parameters, followed by AdaMatch [2] and NSAE [6]. On the other hand, methods like PrototypicalNet [12], OpenMax [1], SnaTCHer [4], OCN [9], MORGAN [8], and DAPN [14], show lower computation complexity than our DAFOS-NET.

Employment of multiple network components like dual-GAN, outlier, and domain predictor makes DAFOS-NET moderately complex with 0.08 GFLOP value and 5.8 million learnable parameters. However, these network components, along with improved metric-learning objectives, help DAFOS-NET acquire significant transferrable knowledge in rejecting $\mathcal{T}$ domain outliers compared to the other methods, as shown in Table 1.

Table 1. Model complexity comparison for 5-way ($\mathcal{K}$) 5-shot (m) evaluation on DomainNet (`Real` to `clipArt`)

Model	GFLOPS	Parameters (M)	AUROC
PrototypicalNet [12]	0.01	3.9	23.43
MetaoptNet [5]	0.7	18.1	30.77
OpenMax [1]	0.03	4.3	17.38
PEELER [7]	0.09	6.2	24.14
SnaTCHer [4]	0.02	4.4	27.16
OCN [9]	0.008	2.3	25.62
MORGAN [8]	0.05	5.2	34.44
AdaMatch [2]	0.1	10.2	31.21
DAPN [14]	0.03	4.8	31.42
NSAE [6]	0.1	9.7	29.32
DAFOS-NET (Ours)	**0.08**	**5.8**	**60.67**

3. Meta-learning of DAFOS-NET

Algorithm 1: DAFOS-NET Meta-training

```
Input: $\mathbb{S}_\mathcal{S}(\mathcal{K}_\mathcal{S})$, $\mathbb{Q}_\mathcal{S}(\mathcal{K}_\mathcal{S} \cup \mathcal{U}_\mathcal{S})$, $\mathbb{S}_\mathcal{T}(\mathcal{K}_\mathcal{T})$,
       $\mathbb{Q}_\mathcal{T}(\mathcal{K}_\mathcal{T} \cup \mathcal{U}_\mathcal{T})$, Networks: $f_\varphi$, $G_{\mathcal{L}\theta}$, $G_{\mathcal{H}\theta}$,
       $D_{\mathcal{L}\phi}$, $D_{\mathcal{H}\phi}$, $\mathcal{O}_\eta$, $\mathcal{O}_\zeta$, Variance: $\sigma L$, $\sigma H$,
       Iterations: $\mathcal{I}$, Learning rates: $\alpha$, $\beta$
Output: Updated model parameters of $f_\varphi$, $G_{\mathcal{L}\theta}$,
        $G_{\mathcal{H}\theta}$, $D_{\mathcal{L}\phi}$, $D_{\mathcal{H}\phi}$, $\mathcal{O}_\eta$, $\mathcal{O}_\zeta$

1 Weak-strong augmentation ($W/S$) of image space:
    $\mathbb{S}_\mathcal{S} \leftarrow \{\mathbb{S}_\mathcal{S}, (\mathbb{S}_\mathcal{S})_{W/S}\}$, $\mathbb{S}_\mathcal{T} \leftarrow \{\mathbb{S}_\mathcal{T}, (\mathbb{S}_\mathcal{T})_{W/S}\}$
2 Extract features: $\{\mathbb{S}_{f\mathcal{S}}, \mathbb{S}_{f\mathcal{T}}\}$=$f(\{\mathbb{S}_\mathcal{S}, \mathbb{S}_\mathcal{T}\}; \varphi)$,
    $\{\mathbb{Q}_{f\mathcal{S}}, \mathbb{Q}_{f\mathcal{T}}\}$=$f(\{\mathbb{Q}_\mathcal{S}, \mathbb{Q}_\mathcal{T}\}; \varphi)$;
3 Update cGANs for each domain ($\mathcal{D}$) alternatively;
  3.1 Randomly initialize $\mathcal{L}\theta$, $\mathcal{L}\phi$, $\mathcal{H}\theta$, $\mathcal{H}\phi$
  for $i \leftarrow 1$ to $|\mathcal{D}|$ do                    // $\mathcal{D} \in \{\mathcal{S}, \mathcal{T}\}$
      3.2 Clone GAN parameters: $D_{\widehat{\mathcal{L}\phi}} \leftarrow D_{\mathcal{L}\phi}$,
          $D_{\widehat{\mathcal{H}\phi}} \leftarrow D_{\mathcal{H}\phi}$, $G_{\widehat{\mathcal{L}\theta}} \leftarrow G_{\mathcal{L}\theta}$, $G_{\widehat{\mathcal{H}\theta}} \leftarrow G_{\mathcal{H}\theta}$ ;
      3.3 Sample episodes: $\tau \sim \mathbb{S}_\mathcal{S}$ if $i == 1$ else $\mathbb{S}_\mathcal{T}$ ;
      3.4 $\nabla DL, \nabla GL, s_l, z_l \leftarrow$ InnerLoop($D_{\widehat{\mathcal{L}\phi}}$,
          $G_{\widehat{\mathcal{L}\theta}}$, $D_{\mathcal{L}\phi}$, $G_{\mathcal{L}\theta}$, $\alpha$, $\mathcal{I}$, $\tau$, $\sigma L$, $0, 0$, $\mathcal{K}/\mathcal{U}$, $\mathcal{D}$);
      3.5 $\nabla DH, \nabla GH, s_h, z_h \leftarrow$ InnerLoop($D_{\widehat{\mathcal{H}\phi}}$,
          $G_{\widehat{\mathcal{H}\theta}}$, $D_{\mathcal{H}\phi}$, $G_{\mathcal{H}\theta}$, $\alpha$, $\mathcal{I}$, $\tau$, $\sigma H$, $s_l$, $z_l$, $\mathcal{K}/\mathcal{U}$, $\mathcal{D}$);
      3.6 Update GAN parameters:
          $\mathcal{H}\phi \leftarrow \mathcal{H}\phi - \beta \times \nabla DH$; $\mathcal{H}\theta \leftarrow \mathcal{H}\theta - \beta \times \nabla GH$;
          $\mathcal{L}\phi \leftarrow \mathcal{L}\phi - \beta \times \nabla DL$; $\mathcal{L}\theta \leftarrow \mathcal{L}\theta - \beta \times \nabla GL$;
      3.7 Assign $\{\mathbb{S}_{l\mathcal{S}}, \mathbb{S}_{h\mathcal{S}}\} \leftarrow \{\mathbb{S}_l, \mathbb{S}_h\}$ if $i == 1$
          else $\{\mathbb{S}_{l\mathcal{T}}, \mathbb{S}_{h\mathcal{T}}\} \leftarrow \{\mathbb{S}_l, \mathbb{S}_h\}$;
4 Augment closed-set, get prototypes $\mathcal{P}_\mathcal{S}$, $\mathcal{P}_\mathcal{T}$ (3)
5 Estimate query distances, $\mathcal{Q}_{dist|(\mathcal{S}\cup\mathcal{T})}$;
6 $\mathcal{Q}_{dist}$ is passed to $\mathcal{O}_\eta$, $\mathcal{O}_\zeta$ to classify each query
    $q \in \{\mathbb{Q}_{f\mathcal{S}}, \mathbb{S}_{h\mathcal{S}}, \mathbb{Q}_{f\mathcal{T}}, \mathbb{S}_{h\mathcal{T}}\}$ $\mathcal{S}/\mathcal{T}$, known/outlier ;
  if ($\mathcal{O}_\zeta \rightarrow$ known) and ($\mathcal{O}_\eta \rightarrow$ Source) then
      Predict $\mathcal{S}$ domain class: Softmax($\mathcal{P}_\mathcal{S} - q$) ;
  else if ($\mathcal{O}_\zeta \rightarrow$ outlier) and ($\mathcal{O}_\eta \rightarrow$ Source) then
      Predict query as $\mathcal{S}$ domain outlier ;
  else if ($\mathcal{O}_\zeta \rightarrow$ known) and ($\mathcal{O}_\eta \rightarrow$ Target) then
      Predict $\mathcal{T}$ domain class: Softmax($\mathcal{P}_\mathcal{T} - q$) ;
  else
      Predict query as $\mathcal{T}$ domain outlier ;
7 Compute metric losses: $\mathcal{L}_{Align}$ (4), $\mathcal{L}_{C|(\mathcal{S}\cup\mathcal{T})}$ (5),
    $\mathcal{L}_{PD}$ (6), $\mathcal{L}_{OUT}$ (7), $\mathcal{L}_{DC}$ (8), and $\mathcal{L}_{FE}$ (9);
8 Optimize $f_\varphi$, $G_{\mathcal{L}\theta}$, $G_{\mathcal{H}\theta}$, $D_{\mathcal{L}\phi}$, $D_{\mathcal{H}\phi}$, $\mathcal{O}_\eta$, $\mathcal{O}_\zeta$.
  return Updated $f_\varphi$, cGANs, $\mathcal{O}_\eta$, $\mathcal{O}_\zeta$;
```

(#) All equation numbers are as per the Main paper.
(*) For 5-shot meta training, we simultaneously take samples from $\mathcal{S}$ and $\mathcal{T}$, where, $m_\mathcal{S} = 3, m_\mathcal{T} = 2$. Whereas, for 1-shot training, we take both domain samples in alternate iterations, where, $m_\mathcal{S} = m_\mathcal{T} = 1$. For both 1 or 5-shot training, we select 15 disjoint query samples from each of $\mathbb{Q}_\mathcal{S}$ and $\mathbb{Q}_\mathcal{T}$.

Algorithm 2: DAFOS-NET Inner Loop update

```
Input: $D_{\widehat{\phi}}$, $G_{\widehat{\theta}}$, $D_\phi$, $G_\theta$, $\alpha$, $\mathcal{I}$,$\tau$, $\sigma$, $s$, $z$, $\mathcal{K}/\mathcal{U}$, $\mathcal{D}$
Output: $\nabla D$, $\nabla G$, $s'$: new adversarial sample, $z'$
1 Randomly draw $K$ Samples $\{s_1, ..., s_K\} \sim \tau$ ;
2 for $i \leftarrow 1$ to $\mathcal{I}$ do
      for $k \leftarrow 1$ to $K$ do
          2.1 $s' \leftarrow G_{\widehat{\theta}}(z', \mathcal{K}/\mathcal{U}, \mathcal{D})$ , $z' \sim \mathcal{N}(0, \sigma)$;
          2.2 Update $D_{\widehat{\phi}}$ and $G_{\widehat{\theta}}$ parameters:
              $\widehat{\phi} \leftarrow \phi - \alpha \times \nabla_{\widehat{\phi}}(L^D_{\widehat{\theta},\widehat{\phi}}(s_k, s') + \mathcal{L}_{AOCMC}$ ;
              $\widehat{\theta} \leftarrow \theta - \alpha \times \nabla_{\widehat{\theta}}(L^G_{\widehat{\theta},\widehat{\phi}}(s') + \mathcal{L}_{AOCMC}(2))$;
      2.3 Compute: $\nabla D \leftarrow \phi - \widehat{\phi}$, $\nabla G \leftarrow \theta - \widehat{\theta}$ ;
  return $\nabla D, \nabla G, s', z'$
```

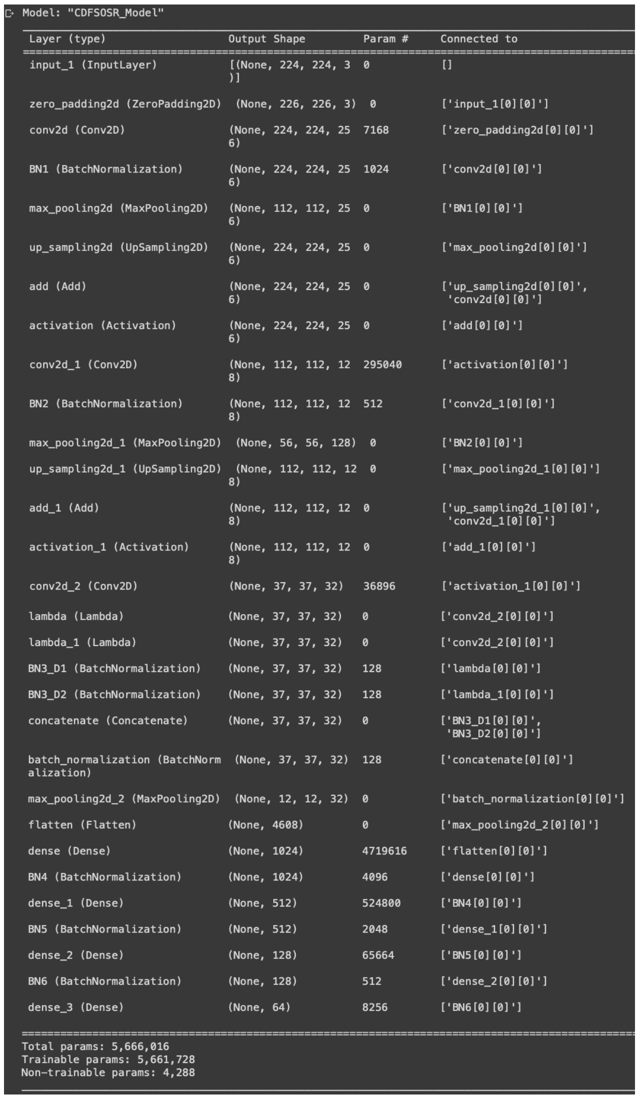
Algorithm 3: DAFOS-NET meta-testing phase for Standard DA-FSOS setting

```
Input: $\mathbb{S}(\mathcal{K})$, $\mathbb{Q}(\mathcal{K} \cup \mathcal{U})$, $f_\varphi$,$\mathcal{O}_\zeta$
Output: Open-set classification of $\mathcal{T}$ query samples
1 $\mathbb{S}_f = f(\mathbb{S}(\mathcal{K}); \varphi)$, $\mathbb{Q}_f = f(\mathbb{Q}(\mathcal{K} \cup \mathcal{U}); \varphi)$;
2 Compute prototype $\mathcal{P}$ from $\mathbb{S}_f$;
3 Measure query distances: $\mathcal{Q}_{dist} \leftarrow \|\mathcal{P} - \mathbb{Q}_f\|_2^2$ ;
4 Classify queries as known/outlier: $\mathcal{O}_\zeta(\mathcal{Q}_{dist})$ ;
  if $\mathcal{O}_\zeta$ predicts known then
      Find $\mathcal{T}$ domain class: Softmax($\mathcal{Q}_{dist}$) ;
  else
      Assign the query as outlier ;
  return Class predictions of $\mathcal{T}$ query samples;
```

Algorithm 4: DAFOS-NET meta-testing phase for Generalized DA-FSOS setting

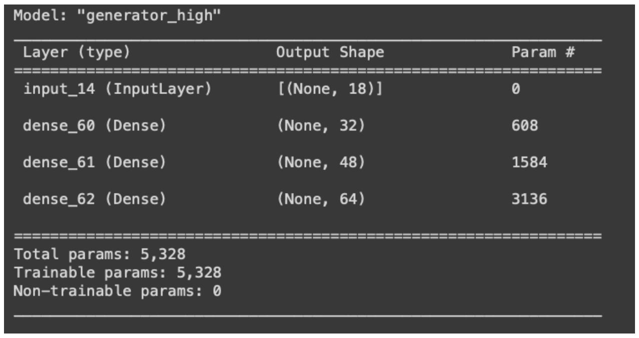
```
Input: $\mathbb{S}(\mathcal{K})$, $\mathbb{Q}(\mathcal{K} \cup \mathcal{U})$, $f_\varphi$, $\mathcal{O}_\eta$, $\mathcal{O}_\zeta$
Output: Open-set classification of $\mathcal{S} + \mathcal{T}$ queries
1 $\mathbb{S}_f = f(\mathbb{S}(\mathcal{K}); \varphi)$, $\mathbb{Q}_f = f(\mathbb{Q}(\mathcal{K} \cup \mathcal{U}); \varphi)$;
2 Compute prototype $\mathcal{P}$ from $\mathbb{S}_f$;
3 Measure query distances: $\mathcal{Q}_{dist} \leftarrow \|\mathcal{P} - \mathbb{Q}_f\|_2^2$ ;
4 Classify queries as known / outlier and $\mathcal{S}/\mathcal{T}$ sample;
  if ($\mathcal{O}_\zeta \rightarrow$ known) and ($\mathcal{O}_\eta \rightarrow$ Source) then
      $\mathcal{P}_\mathcal{S} \leftarrow \mathcal{P}$, Predict class: Softmax($\mathcal{P}_\mathcal{S} - q$) ;
  else if ($\mathcal{O}_\zeta \rightarrow$ outlier) and ($\mathcal{O}_\eta \rightarrow$ Source) then
      Predict query as $\mathcal{S}$ domain outlier ;
  else if ($\mathcal{O}_\zeta \rightarrow$ known) and ($\mathcal{O}_\eta \rightarrow$ Target) then
      $\mathcal{P}_\mathcal{T} \leftarrow \mathcal{P}$, Predict class: Softmax($\mathcal{P}_\mathcal{T} - q$) ;
  else
      Predict query as $\mathcal{T}$ domain outlier ;
  return Class predictions of $\mathcal{S} + \mathcal{T}$ query samples;
```

4. DAFOS-NET model parameters



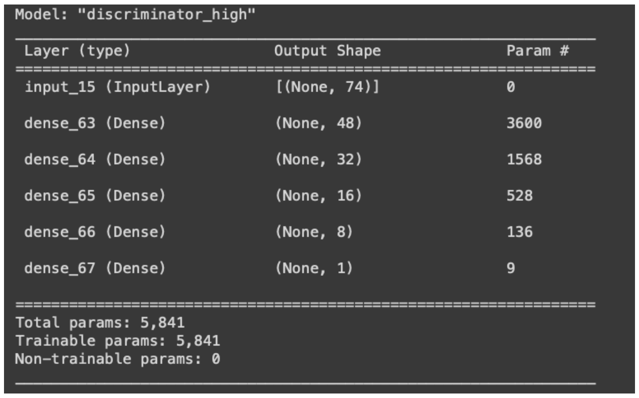
```
Model: "CDFSOSR_Model"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_1 (InputLayer)           [(None, 224, 224, 3  0           []
                                )]

 zero_padding2d (ZeroPadding2D)  (None, 226, 226, 3)  0          ['input_1[0][0]']

 conv2d (Conv2D)                (None, 224, 224, 25  7168        ['zero_padding2d[0][0]']
                                6)

 BN1 (BatchNormalization)       (None, 224, 224, 25  1024        ['conv2d[0][0]']
                                6)

 max_pooling2d (MaxPooling2D)   (None, 112, 112, 25  0           ['BN1[0][0]']
                                6)

 up_sampling2d (UpSampling2D)   (None, 224, 224, 25  0           ['max_pooling2d[0][0]']
                                6)

 add (Add)                      (None, 224, 224, 25  0           ['up_sampling2d[0][0]',
                                6)                                'conv2d[0][0]']

 activation (Activation)        (None, 224, 224, 25  0           ['add[0][0]']
                                6)

 conv2d_1 (Conv2D)              (None, 112, 112, 12  295040      ['activation[0][0]']
                                8)

 BN2 (BatchNormalization)       (None, 112, 112, 12  512         ['conv2d_1[0][0]']
                                8)

 max_pooling2d_1 (MaxPooling2D)  (None, 56, 56, 128)  0          ['BN2[0][0]']

 up_sampling2d_1 (UpSampling2D)  (None, 112, 112, 12  0          ['max_pooling2d_1[0][0]']
                                8)

 add_1 (Add)                    (None, 112, 112, 12  0           ['up_sampling2d_1[0][0]',
                                8)                                'conv2d_1[0][0]']

 activation_1 (Activation)      (None, 112, 112, 12  0           ['add_1[0][0]']
                                8)

 conv2d_2 (Conv2D)              (None, 37, 37, 32)   36896       ['activation_1[0][0]']

 lambda (Lambda)                (None, 37, 37, 32)   0           ['conv2d_2[0][0]']

 lambda_1 (Lambda)              (None, 37, 37, 32)   0           ['conv2d_2[0][0]']

 BN3_D1 (BatchNormalization)    (None, 37, 37, 32)   128         ['lambda[0][0]']

 BN3_D2 (BatchNormalization)    (None, 37, 37, 32)   128         ['lambda_1[0][0]']

 concatenate (Concatenate)      (None, 37, 37, 32)   0           ['BN3_D1[0][0]',
                                                                  'BN3_D2[0][0]']

 batch_normalization (BatchNorm  (None, 37, 37, 32)  128         ['concatenate[0][0]']
 alization)

 max_pooling2d_2 (MaxPooling2D)  (None, 12, 12, 32)  0           ['batch_normalization[0][0]']

 flatten (Flatten)              (None, 4608)         0           ['max_pooling2d_2[0][0]']

 dense (Dense)                  (None, 1024)         4719616     ['flatten[0][0]']

 BN4 (BatchNormalization)       (None, 1024)         4096        ['dense[0][0]']

 dense_1 (Dense)                (None, 512)          524800      ['BN4[0][0]']

 BN5 (BatchNormalization)       (None, 512)          2048        ['dense_1[0][0]']

 dense_2 (Dense)                (None, 128)          65664       ['BN5[0][0]']

 BN6 (BatchNormalization)       (None, 128)          512         ['dense_2[0][0]']

 dense_3 (Dense)                (None, 64)           8256        ['BN6[0][0]']

==================================================================================================
Total params: 5,666,016
Trainable params: 5,661,728
Non-trainable params: 4,288
__________________________________________________________________________________________________
```

Figure 1. The layer-wise summary of the feature extractor f_φ following ResNet-18 [3] as backbone along with our domain-specific batch-norm layers incorporated in between the final convolution layer blocks of f_φ



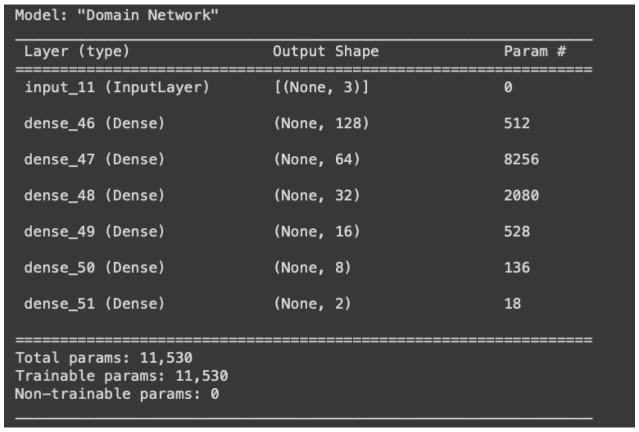
```
Model: "generator_high"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_14 (InputLayer)       [(None, 18)]              0

 dense_60 (Dense)            (None, 32)                608

 dense_61 (Dense)            (None, 48)                1584

 dense_62 (Dense)            (None, 64)                3136

=================================================================
Total params: 5,328
Trainable params: 5,328
Non-trainable params: 0
_________________________________________________________________
```

Figure 2. The layer-wise summary of $G_{\mathcal{H}\theta}$. An identical structure is followed for $G_{\mathcal{L}\theta}$.



```
Model: "discriminator_high"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_15 (InputLayer)       [(None, 74)]              0

 dense_63 (Dense)            (None, 48)                3600

 dense_64 (Dense)            (None, 32)                1568

 dense_65 (Dense)            (None, 16)                528

 dense_66 (Dense)            (None, 8)                 136

 dense_67 (Dense)            (None, 1)                 9

=================================================================
Total params: 5,841
Trainable params: 5,841
Non-trainable params: 0
_________________________________________________________________
```

Figure 3. The layer-wise summary of $D_{\mathcal{H}\phi}$. An identical structure is followed for $D_{\mathcal{L}\phi}$.



```
Model: "Domain Network"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_11 (InputLayer)       [(None, 3)]               0

 dense_46 (Dense)            (None, 128)               512

 dense_47 (Dense)            (None, 64)                8256

 dense_48 (Dense)            (None, 32)                2080

 dense_49 (Dense)            (None, 16)                528

 dense_50 (Dense)            (None, 8)                 136

 dense_51 (Dense)            (None, 2)                 18

=================================================================
Total params: 11,530
Trainable params: 11,530
Non-trainable params: 0
_________________________________________________________________
```

Figure 4. The layer-wise summary of $\mathcal{O}_\eta$. An identical structure is followed for $\mathcal{O}_\zeta$.

5. AUROC plot Analysis

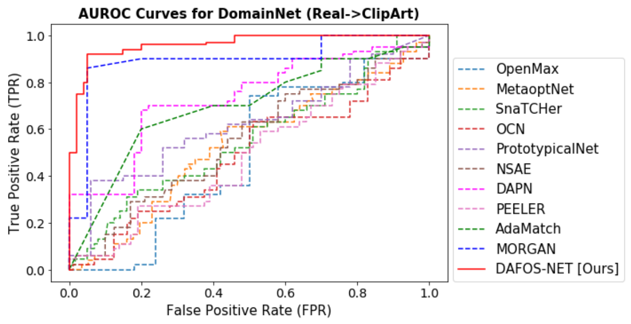


Figure 5. 5-way 5-shot AUROC comparison of different methods on DomainNet (`Real` to `clipart`). Our DAFOS-NET (represented in 'Red' color) shows the highest True-Positive Rate performance for a low False-Positive Rate value over all other methods

We evaluate the outlier rejection capability of DAFOS-NET and compare it against different methods in terms of Area under Receiver Operating Characteristics Curve (AUROC) in Fig. 5. The figure reveals that DAFOS-NET envelops the maximum area under the ROC curve in contrast to the baseline methods, indicating a very high outlier detection performance under the challenging domain-shift and few-shot difficulty problems. Experimental analysis in Table 1 of the Main paper also advocates the same. Overall, the superior AUROC performance by DAFOS-NET sets it as the baseline DA-FSOS method.

6. Few-shot closed Set comparison

Table 2. Few-shot closed accuracy (Acc) comparison of different methods based on DA-FSL protocol for evaluation over DomainNet (`Real` to `clipart`) [5-way 1/5 shot]

Method	1-shot	5-shot
AdaMatch [2]	35.18	42.29
DAPN [14]	37.21	54.47
NSAE [6]	36.75	49.32
DAFOS-NET (ours)	**55.26**	**60.35**

In Table 2, we compare the performance of DAFOS-NET against a state-of-the-art (SOTA) domain adaptation method, AdaMatch, and two DA-FSL methods, NSAE and DAPN, under the common DA-FSL experimental protocol. According to the DA-FSL setting, there is no outlier during the meta-testing phase. Hence, during the meta-training phase, we also did not consider any pseudo-unknown class samples to learn to reject them. We find a significant 1-shot performance boost, approximately 17% over the literature, by our method. Thanks to the batch-norm-based global cross-domain prototype alignment and Known-class compactness loss in optimizing DAFOS-NET, target domain queries reside close to their respective prototypes. Thereby, it boosts overall closed-set classification accuracy.

7. Impact of Noise variance

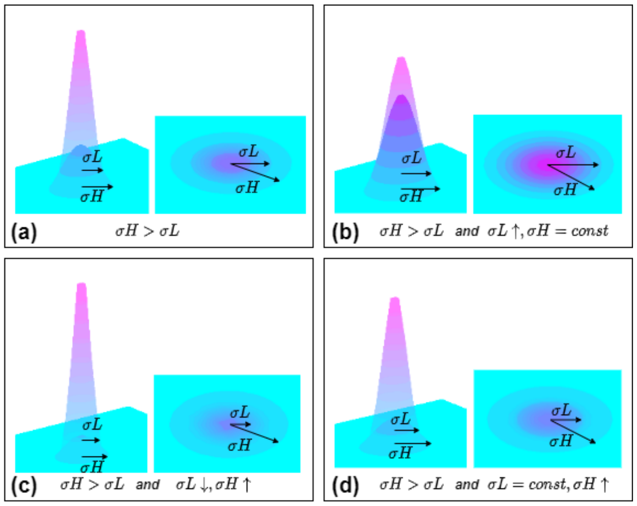

σL
σH
(a) $\sigma H > \sigma L$
(b) $\sigma H > \sigma L$ and $\sigma L \uparrow, \sigma H = const$
(c) $\sigma H > \sigma L$ and $\sigma L \downarrow, \sigma H \uparrow$
(d) $\sigma H > \sigma L$ and $\sigma L = const, \sigma H \uparrow$

Figure 6. Impact of two noise vectors (z_l, z_h) with varying standard deviations $(\sigma L, \sigma H)$ on two adversarial generators $G_{\mathcal{L}\theta}, G_{\mathcal{H}\theta}$ producing pseudo-known and unknown samples, when (b) σL is increasing while σH is constant (b) σL is decreasing when σH is increasing, and (c) σL is constant with rising σH. The generated adversarial samples used for augmentation create consequences in the model's closed-open space sample recognition performance. In all the cases, we follow an additional common criterion as in a) $\sigma H > \sigma L$.

Over three scenarios in Fig. 6, we investigate the consequences of pseudo-known and unknown sample generation on the model's known and outlier recognition ability. It is worth mentioning that pseudo-known samples are generated from an isotropic Gaussian noise vector $z_l \in \mathcal{N}(0, \sigma L)$ and pseudo-unknown samples are produced from another noise vector $z_h \in \mathcal{N}(0, \sigma H)$. Ideally, pseudo-unknown samples should be generated from a subset of $\{\mathcal{N}(0, \sigma H) - \mathcal{N}(0, \sigma L)\}$ region.

Case-1 $\sigma L \uparrow, \sigma H = constant$ (Fig 6b.): The pseudo-known samples enrich the closed-set data density, but it produces limited discriminative pseudo unknown samples from reduced $\{\mathcal{N}(0, \sigma H) - \mathcal{N}(0, \sigma L)\}$ region. Thereby, accuracy increases, but AUROC decreases.

Case-2 $\sigma L \downarrow, \sigma H \uparrow$ (Fig 6c.): The model lacks a diversity of closed-set samples whereas open space is populated with many pseudo-unknown samples. Also, many pseudo-known samples erroneously get detected as outliers. It impacts reduced accuracy and increased AUROC values.

Case-3 $\sigma L = constant, \sigma H \uparrow$ (Fig 6d.): Beyond a certain limit, increasing σH causes pseudo-unknown samples to mix with other known class samples, reducing AUROC.

Based on the above three scenarios, we perform ablation studies and experimentally find in Table 5 (Main paper) that the combination of $\sigma L = 0.3, \sigma H = 0.9$ provide the optimal performance.

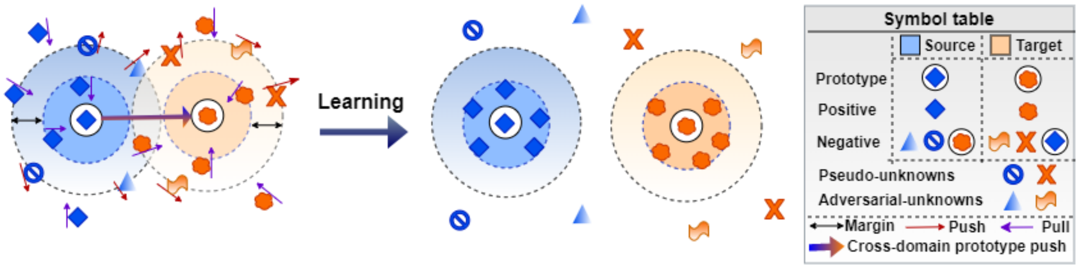


Figure 7. Prototype diversification loss pulls the positive class queries to its respective prototype whereas pushes the negative samples, including the pseudo-unknowns with adversarially generated unknown samples of the same domain and cross-domain prototypes, to spread away beyond a closed-open separation margin.

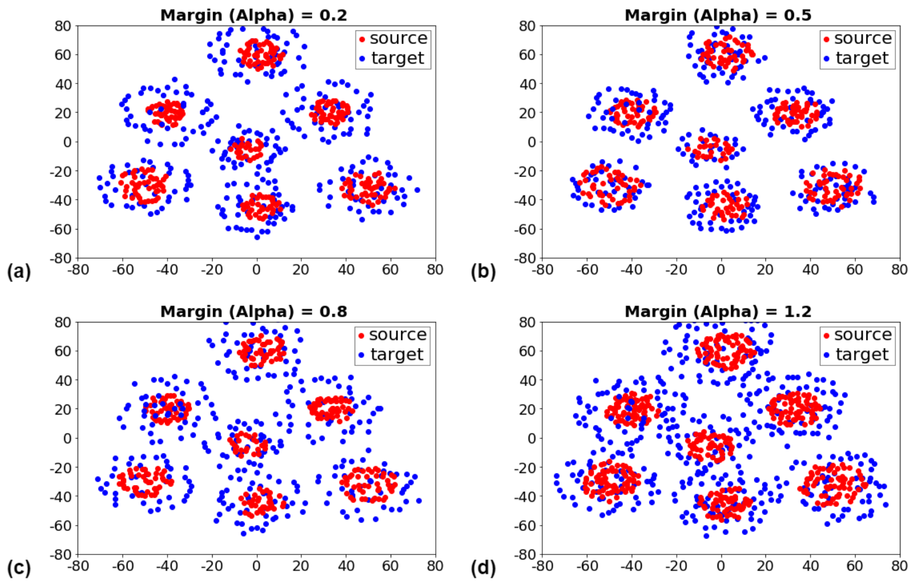


Figure 8. The t-SNE visualization of the metric space due to optimization by the Prototype Diversification Loss with varying margin (α) values, (a) $\alpha = 0.2$, (b) $\alpha = 0.5$, (c) $\alpha = 0.8$, and (d) $\alpha = 1.2$. Selection of $\alpha = 0.5$ produces the optimal separation of known classes. For lower value, $\alpha = 0.2$, even though source samples form dense clusters, target known class samples fail to adapt to dense compaction. Again, increasing $\alpha > 0.5$, we observe the sparsity of source samples and a mixture of target samples to other known classes.

8. Prototype diversification loss

i. Visual illustration: In Fig 7, we illustrate the working principle of our Prototype Diversification Loss, $\mathcal{L}_{PD}$ (Eq. 6 in Main paper). In order to construct a domain-agnostic firm decision boundary separating the closed-open space, only rejecting the outliers does not satisfy the necessary and sufficient condition. It becomes essential to push these outliers beyond a certain margin. Effectively, the margin separation helps to discriminate the fine-grained outliers and thereby improves AUROC performance. To comprehend the same, we develop $\mathcal{L}_{PD}$ so that a prototype can pull its positive class in-domain queries while simultaneously pushing all the negative queries comprised of in-domain pseudo-unknowns, adversarial unknowns, and cross-domain prototypes beyond certain margin value, α. Our notion is to form a domain-invariant bounded closed space while maintaining discriminativeness and diversity across domains; an unbounded open space surrounds this closed space.

ii. Analysis of margin value: We vary α hyper-parameter value and monitor its impact through t-SNE visualization in Fig 8. For ease of understanding, we only select seven classes, namely Axe, Bucket, Bus, Clock, Flower, Foot, and Strawberry, across the source and target domains from a wide range of 345 classes of DomainNet.

Learning with a very low margin value, i.e., $\alpha = 0.2$, reduces the intra-class distance and increases the inter-class separation of source samples, but target samples fail to adapt compaction at a similar extent, impacting reduced closed-set accuracy for the target domain. Again, at $\alpha = 0.8$, the source domain samples become sparse while the target domain known class samples tend to diffuse to other negative class prototypes. This phenomenon becomes extreme when $\alpha > 1$, it becomes very tough to distinguish known and pseudo-unknown samples, impacting both accuracy and AUROC. We find the optimum value for $\alpha = 0.5$, balancing both the metrics for source and target domain.

Table 3. The 5-way 5-shot DA-FSOS performance comparison of DAFOS-Net and other Methods over Office-Home dataset

Model	Venue	Paradigm	Real-World to Art		Real-World to Product	
			Acc(%)	AUROC(%)	Acc(%)	AUROC(%)
PrototypicalNet [11]	NIPS-17	FSL	22.11	12.22	20.78	10.62
Metaoptnet [5]	CVPR-19	FSL	29.31	18.14	28.42	16.33
OpenMax[1]	CVPR-16	OSR	13.96	14.05	12.61	11.15
PEELER [7]	CVPR-20	FSOS	14.54	15.92	13.25	13.69
SnaTCHer [4]	CVPR-21	FSOS	21.71	22.66	19.72	18.67
OCN [9]	WACV-22	FSOS	23.34	26.49	20.41	22.31
MORGAN [8]	WACV-23	FSOS	31.83	33.35	30.29	32.25
AdaMatch [2]	ICLR-22	DA	27.24	15.44	25.57	13.14
DAPN [14]	WACV-21	DA-FSL	29.17	18.67	27.66	17.36
NSAE [6]	ICCV-21	CDFSL	32.16	20.24	29.31	17.23
MORGAN + DAPN	-	-	36.52	31.32	33.63	32.27
DAFOS-Net [Ours]	-	DA-FSOS	**56.74±0.73**	**53.42±0.68**	**61.70±0.86**	**48.41±0.59**

Table 4. The 5-way 5-shot DA-FSOS performance comparison of DAFOS-Net and other Methods over DomainNet dataset

Model	Venue	Paradigm	Real to Quickdraw		Real to Infograph		Real to Sketch	
			Acc(%)	AUROC(%)	Acc(%)	AUROC(%)	Acc(%)	AUROC(%)
PrototypicalNet [11]	NIPS-17	FSL	25.81	11.72	27.63	12.12	30.18	14.15
Metaoptnet [5]	CVPR-19	FSL	31.45	14.51	33.18	17.28	37.65	21.78
OpenMax[1]	CVPR-16	OSR	15.21	14.55	16.44	17.23	18.54	17.44
PEELER [7]	CVPR-20	FSOS	20.25	21.37	21.67	22.88	23.42	22.68
SnaTCHer [4]	CVPR-21	FSOS	21.63	22.66	24.74	21.06	29.25	27.62
OCN [9]	WACV-22	FSOS	20.42	21.24	23.25	25.71	26.89	27.31
MORGAN [8]	WACV-23	FSOS	30.63	32.93	33.59	36.24	37.45	35.34
AdaMatch [2]	ICLR-22	DA	30.57	15.31	31.89	17.17	37.53	20.62
DAPN [14]	WACV-21	DA-FSL	29.88	17.46	34.68	20.26	39.74	24.58
NSAE [6]	ICCV-21	CDFSL	30.06	19.32	33.27	23.53	35.36	26.21
MORGAN + DAPN	-	-	31.93	34.24	35.51	37.64	38.38	36.69
DAFOS-Net [Ours]	-	DA-FSOS	**59.92±0.58**	**57.93±0.44**	**57.54±0.62**	**58.37±0.37**	**59.72±0.46**	**63.65±0.81**

9. DA-FSOS comparison results

We show the DA-FSOS performance comparison result of the proposed DAFOS-Net against the SOTA models over Office-Home (`Real-World` to `Clipart`) and DomainNet ((a) `Real` to `Clipart`, b) `Real` to `Painting`, and c) `Clipart` to `Painting`) datasets in Table 1 of the Main paper. Here we present the comparison results over the remaining domains for Office-Home [13] and DomainNet [10] in Table 3 and 4, respectively.

Training and Evaluation protocol of all methods: During meta-training, FSL methods, PrototypicalNet [11], and Metaoptnet [5] do not get any pseudo-unknown class samples in the query set. Hence, they do not get transferable knowledge on rejecting outliers in test time. Moreover, these methods are trained with only source domain samples and do not learn to handle domain shifts in training time. At test time, they suddenly encounter target domain samples with a mixture of known and outliers. Hence, fail to recognize target outliers impacting heavy drops in AUROC.

OpenMax[1], a post-training outlier calibration method with large-scale data, gets limited samples from the source domain to calibrate its Weibull tails during the training phase. However, It succumbs to distinguishing outliers from a different domain as the Weibull models can not adapt different domain data in test time.

FSOS methods, namely PEELER [7], SnaTCHer [4], OCN [9], MORGAN [8] get only source domain known and pseudo-unknown samples to learn to reject outliers under few-shot difficulty. Nevertheless, their performance dropped in the testing phase to tackle different domain mixed samples of known and outliers for classification.

On the other hand, naive domain-adaptation method, AdaMatch [2] and DA-FSL methods, DAPN [14], NSAE [6] greatly perform to recognize known class samples from target domain but they do not get exposure to pseudo-unknown samples during training. Effectively, while testing, when they encounter target samples with marginal closed-open separation, many known samples get recognized as outliers and vice-versa.

The best performing FSOS baseline method, MORGAN, coupled with *Domain Adaptive Loss* from DAPN [14] during training, receives both domain known and pseudo-unknown samples following the DA-FSOS protocol, shows much better performance in terms of both metrics compared to their singular counterparts. Notably, the poor adaptation scheme of MORGAN+DAPN compared to our batch-norm aware global prototype alignment causes reduced accuracy and AUROC compared to DAFOS-Net.

Finally, we meta-train DAFOS-Net with standard DA-FSOS protocol having both domain known and pseudo-unknown class samples and test time only target samples, as mentioned in (Sec: 3). In all cases, we observe DAFOS-Net achieves a notable performance gain over the literature in terms of closed-set accuracy (Acc) and outlier rejecting AUROC values. The optimal combination of novel metric losses become the ultimate differentiating factor in beating the baseline methods with a wide margin.

Outlier detection mechanism: It is worth mentioning that OCN [9], MORGAN [8], and our DAFOS-NET are already equipped with a binary classification network for rejecting outliers. OpenMax[1] applies a threshold value of 90% on the Weibull cumulative distribution function (CDF) probability where the Weibull models are fit with activation vectors due to pre-training by known class data. We apply confidence thresholding (80% threshold) for the remaining methods to detect outliers.

10. Anti open close mode collapse Regularizer:

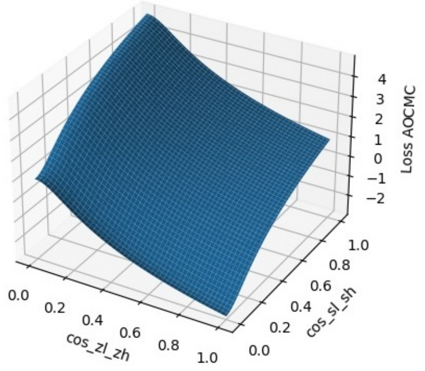


Figure 9. The 3D surface plot of $\mathcal{L}_{AOCMC}$ regularizer due to the similarity between input noise vectors z_l, z_h and the similarity between generated samples s_l, s_h.

In Section 3 of the Main paper, we propose a novel loss function, Anti Open Close Mode Collapse (AOCMC) loss, to discourage pseudo-unknown-sample generators from producing discriminative samples from generated pseudo-known samples. The effect of the similarity between the input noise vectors and the generated pseudo-known-unknown samples on the $\mathcal{L}_{AOCMC}$ loss value is visualized in Fig. 9. We incorporate a small coefficient, $\epsilon = 0.01$, for numerical stability while defining $\mathcal{L}_{AOCMC}$. When the similarity of input noise vectors, $\cos(z_l, z_h)$, is very low, the regularization effect decreases if the similarity of generated pseudo-known and unknown samples, $\cos(s_l, s_h)$, is also low. However, if the similarity between the generated pseudo-known and unknown samples $\cos(s_l, s_h)$ increases, the potential impact of mode-collapse also increases significantly. In such cases, a high regularization value of $\mathcal{L}_{AOCMC}$ is enforced.

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, 6, 7

[2] David Berthelot, Rebecca Roelofs, Kihyuk Sohn, Nicholas Carlini, and Alex Kurakin. Adamatch: A unified approach to semi-supervised learning and domain adaptation. *arXiv preprint arXiv:2106.04732*, 2021. 1, 4, 6

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 3

[4] Minki Jeong, Seokeon Choi, and Changick Kim. Few-shot open-set recognition by transformation consistency. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12566–12575, 2021. 1, 6

[5] Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, and Stefano Soatto. Meta-learning with differentiable convex optimization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10657–10665, 2019. 1, 6

[6] Hanwen Liang, Qiong Zhang, Peng Dai, and Juwei Lu. Boosting the generalization capability in cross-domain few-shot learning via noise-enhanced supervised autoencoder. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9424–9434, 2021. 1, 4, 6

[7] 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. 1, 6

[8] Debabrata Pal, Shirsha Bose, Biplab Banerjee, and Yogananda Jeppu. Morgan: Meta-learning-based few-shot open-set recognition via generative adversarial network. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 6295–6304, 2023. 1, 6, 7

[9] Debabrata Pal, Valay Bundele, Renuka Sharma, Biplab Banerjee, and Yogananda Jeppu. Few-shot open-set recognition of hyperspectral images with outlier calibration network. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3801–3810, 2022. 1, 6, 7

[10] Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1406–1415, 2019. 6

[11] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. *Advances in neural information processing systems*, 30, 2017. 6

[12] Jake Snell, Kevin Swersky, and Richard S Zemel. Prototypical networks for few-shot learning. *arXiv preprint arXiv:1703.05175*, 2017. 1

[13] Hemanth Venkateswara, Jose Eusebio, Shayok Chakraborty, and Sethuraman Panchanathan. Deep hashing network for unsupervised domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5018–5027, 2017. 6

[14] An Zhao, Mingyu Ding, Zhiwu Lu, Tao Xiang, Yulei Niu, Jiechao Guan, and Ji-Rong Wen. Domain-adaptive few-shot learning. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1390–1399, 2021. 1, 4, 6