On the Effectiveness of Image Rotation for Open Set Domain Adaptation Supplementary Material

Silvia Bucci*,1[0000−0001−6318−7288], Mohammad Reza Loghmani*,2[0000−0002−2687−7877], and Tatiana Tommasi1[0000−0001−8229−7159]

[1] Italian Institute of Technology, Italy
[2] Politecnico di Torino, Italy
{silvia.bucci,tatiana.tommasi}@polito.it
[3] Vision for Robotics laboratory, ACIN, TU Wien, 1040 Vienna, Austria
loghmani@acin.tuwien.ac.at

1 Implementation Details

In this section we provide all the implementation details about our method ROS and the parameters used in running all our experiments. We ran ROS on Office-31 [11], Office-Home [14] and using two different backbones, ResNet-50 [6] and VGGNet [13]. For an overall scheme of our architecture, we refer the reader to Figure 1 of the main paper.

Encoder E, ResNet-50 : it is composed by all the layers of a standard ResNet-50 up to the average pooling layer. We start from the encoder model pre-trained on ImageNet [3] and we update only the last convolutional block, finetuning it with learning rate 0.0003.

Classifiers C_1, C_2, ResNet-50 : they are both mainly composed by two Fully Connected (FC) layers. Specifically the first FC has output 256 and is followed by a Batch Normalization [8] layer and Leakly-ReLU (with negative slope angle as 0.2). The second FC changes depending on the classifier: for C_1 it has $|\mathcal{C}_s|$ outputs, while for C_2 it has $|\mathcal{C}_s|+1$ outputs including the unknown category. All the layers are learned from scratch with learning rate 0.003.

Rotation classifiers R_1, R_2, ResNet-50 : they both have the same structure of the classifiers described above. The only difference is in the number of outputs which is $4 \times |\mathcal{C}_s|$ for R_1 and 4 for R_2. All the layers are learned from scratch with learning rate 0.003.

Stage I and Stage II, ResNet-50 : The network trained in Stage I is used as starting point for Stage II, and we know that for the semantic classifier the set of categories increases by one. To take it into consideration, in Stage II we set the learning rate of the new unknown class to twice that of the known classes (already learned in Stage I).

* equal contributions

Table 1s. Office-Home Resnet50

Office-Home									
	Pr → Rw				Pr → Cl				
	OS	OS*	UNK	**HOS**	OS	OS*	UNK	**HOS**	
STA_{sum} [9]	77.5±2.3	78.1±2.2	63.3±8.9	69.7±5.9	45.7±2.1	44.7±1.9	71.5±6.6	55.0±3.3	
STA_{max} [9]	75.7±2.4	76.2±2.2	64.3±9.8	69.5±6.2	45.1±4.5	44.2±4.7	67.1±1.9	53.2±2.8	
OSBP [12]	76.0±1.3	76.2±1.3	71.7±1.6	73.9±1.4	45.3±1.1	44.5±1.2	66.3±1.8	53.2±0.8	
UAN [17]	81.0±0.1	84.0±0.1	0.1±0.0	0.2±0.0	57.1±0.2	59.1±0.8	0.0±0.0	0.0±0.0	
ROS	71.1±1.3	70.8±1.4	78.4±1.8	**74.4±0.1**	47.5±0.9	46.5±1.0	71.2±0.9	**56.3±0.5**	
	Pr → Ar				Ar → Pr				
	OS	OS*	UNK	**HOS**	OS	OS*	UNK	**HOS**	
STA_{sum}	56.1±2.8	55.4±2.5	73.7±12.0	63.1±5.6	68.4±3.4	68.7±3.8	59.7±5.5	63.7±1.4	
STA_{max}	54.9±6.3	54.2±6.4	72.4±7.6	61.9±6.3	67.3±1.0	68.0±1.8	48.4±22.6	54.0±17.5	
OSBP	59.4±0.7	59.1±0.8	68.1±0.8	**63.2±0.2**	71.3±0.5	71.8±0.5	59.8±0.5	65.2±0.4	
UAN	78.1±0.1	81.1±0.2	0.0±0.0	0.0±0.0	78.1±0.1	81.1±0.2	0.0±0.0	0.0±0.0	
ROS	57.6±0.8	57.3±0.8	64.3±1.7	60.6±1.2	68.4±1.1	68.4±1.2	70.3±1.6	**69.3±0.2**	
	Ar → Rw				Ar → Cl				
	OS	OS*	UNK	**HOS**	OS	OS*	UNK	**HOS**	
STA_{sum}	80.0±0.6	81.1±0.4	50.5±6.3	62.1±5.0	51.3±2.5	50.8±2.7	63.4±3.4	56.3±1.2	
STA_{max}	77.9±0.4	78.6±0.4	60.4±1.9	68.3±1.2	47.0±5.8	46.0±6.3	72.3±6.2	55.8±2.6	
OSBP	78.8±0.8	79.3±0.9	67.5±0.3	72.9±0.5	50.7±0.7	50.2±0.9	61.1±0.7	55.1±0.4	
UAN	85.1±0.1	88.2±0.2	0.1±0.0	0.2±0.0	60.0±0.1	62.4±0.1	0.0±0.0	0.0±0.0	
ROS	75.9±1.1	75.8±1.2	77.2±0.9	**76.5±0.4**	51.5±1.0	50.6±1.1	74.1±1.0	**60.1±0.4**	
	Rw → Ar				Rw → Pr				
	OS	OS*	UNK	**HOS**	OS	OS*	UNK	**HOS**	
STA_{sum}	67.7±2.8	67.9±2.8	62.3±2.8	65.0±2.8	77.1±1.7	77.9±1.7	58.0±4.5	66.4±3.3	
STA_{max}	67.5±1.7	67.5±1.8	66.7±3.0	67.1±1.2	76.3±0.4	77.1±0.5	55.4±1.5	64.5±1.0	
OSBP	66.1±0.5	66.1±0.6	67.3±0.6	66.7±0.4	76.0±0.7	76.3±0.7	68.6±2.1	72.3±1.3	
UAN	74.8±0.0	77.5±0.1	0.1±0.0	0.2±0.0	82.1±0.1	85.0±0.2	0.1±0.1	0.1±0.1	
ROS	67.1±1.0	67.0±1.1	70.8±2.0	**68.8±0.6**	72.2±0.7	72.0±0.6	80.0±1.1	**75.7±0.9**	
	Rw → Cl				Cl → Rw				
	OS	OS*	UNK	**HOS**	OS	OS*	UNK	**HOS**	
STA_{sum}	51.7±3.0	51.4±3.3	57.9±5.3	54.2±0.9	69.5±2.7	69.8±2.4	63.2±8.9	66.3±5.9	
STA_{max}	50.4±2.5	49.9±2.9	61.1±9.8	54.5±2.6	67.0±3.0	67.0±2.8	66.7±8.7	66.8±5.7	
OSBP	48.5±0.5	48.0±0.5	63.0±0.6	54.5±0.2	71.9±0.9	72.0±0.9	69.2±0.2	**70.6±0.4**	
UAN	63.5±0.1	66.2±0.5	0.0±0.0	0.0±0.0	77.7±0.4	80.6±0.4	0.1±0.0	0.2±0.0	
ROS	52.3±0.9	51.5±0.9	73.0±0.8	**60.4±0.5**	65.6±0.3	65.3±0.3	72.2±1.6	68.6±0.7	

	Cl → Ar				Cl → Pr				Avg.			
	OS	OS*	UNK	**HOS**	OS	OS*	UNK	**HOS**	OS	OS*	UNK	**HOS**
STA_{sum}	53.5±3.0	53.0±3.1	63.9±1.2	57.9±2.3	61.5±2.4	61.4±2.4	63.5±3.3	62.5±2.8	63.3±2.1	63.4±2.1	62.6±2.3	61.9±2.1
STA_{max}	52.0±2.2	51.4±2.3	65.0±2.2	57.4±0.7	61.7±1.6	61.8±1.7	59.1±1.1	60.4±0.7	61.9±2.4	61.8±2.6	63.3±1.9	61.1±0.3
OSBP	59.8±0.4	59.4±0.5	70.3±1.3	**64.3±0.4**	66.9±1.5	67.0±1.6	62.7±2.3	64.7±0.7	64.2±0.1	64.1±0.1	66.3±0.4	64.7±0.2
UAN	67.8±0.3	70.5±0.5	0.0±0.0	0.0±0.0	71.3±0.1	74.0±0.2	0.1±0.0	0.2±0.0	72.5±0.0	75.2±0.1	0.0±0.0	0.1±0.0
ROS	54.1±1.0	53.6±1.1	65.5±2.8	58.9±0.5	60.3±0.4	59.8±0.4	71.6±1.1	**65.2±0.7**	62.0±0.2	61.6±0.2	72.4±0.8	**66.2±0.3**

Encoder E, VGGNet : it is composed by all the layers of a standard VGG-19 up to the second fully connected layer. We start from the encoder model pre-trained on ImageNet [3] and we update only the last two FC layers, finetuning it with learning rate 0.0003.

Classifiers C_1, C_2, R_1, R_2, VGGNet : they have exactly the same structure used for the ResNet-50 case described above.

Stage I and Stage II, VGGNet : The network trained in Stage I is not used as starting point for Stage II. Still we consider the learning rate of the extra unknown class in Stage II higher with respect to the other classes (1.5), but lower than the value used in case of ResNet-50 (2), where Stage II was inheriting the model of Stage I. We also tried to inherit the Stage I model for Stage II as done in the ResNet case, but for VGG that setting produced lower results.

Office-31, ResNet-50 : batch size 32, learning rate defined as specified above and decreasing during training with inverse decay scheduling. We used SGD with momentum, setting the weight decay as 0.0005 and momentum as 0.9. The

Table 2s. Office-31

Office-31 ResNet-50												
	A → W				A → D							
	OS	OS*	UNK	**HOS**	OS	OS*	UNK	**HOS**				
STA_{sum} [9]	89.0±4.0	92.1±4.6	58.0±5.7	71.0±4.0	90.8±2.6	95.4±2.8	45.5±1.6	61.6±1.7				
STA_{max} [9]	85.0±4.8	86.7±5.4	67.6±1.3	75.9±1.3	88.5±2.1	91.0±2.6	63.9±3.8	75.0±2.2				
OSBP [12]	86.0±1.1	86.8±1.2	79.2±0.4	**82.7±0.6**	89.2±0.4	90.5±0.4	75.5±1.4	**82.4±0.9**				
UAN [17]	89.4±0.4	95.5±0.1	31.0±0.9	46.8±1.0	89.5±0.4	95.6±0.5	24.4±0.9	38.9±1.1				
ROS	87.3±1.5	88.4±1.7	76.7±2.4	82.1±1.4	86.7±0.5	87.5±0.6	77.8±0.6	**82.4±0.6**				
	D → W				W → D							
	OS	OS*	UNK	**HOS**	OS	OS*	UNK	**HOS**				
STA_{sum}	92.8±1.3	97.1±0.8	49.7±8.1	65.5±7.0	92.2±0.5	96.6±0.4	48.5±6.0	64.4±5.1				
STA_{max}	90.6±2.8	94.1±3.2	55.5±1.3	69.8±0.2	83.4±6.4	84.9±7.2	67.8±5.0	75.2±3.6				
OSBP	97.5±0.5	97.7±0.2	96.7±2.7	**97.2±1.4**	97.8±1.1	99.1±1.0	84.2±2.2	91.1±1.6				
UAN	95.5±0.1	99.8±0.0	52.5±1.1	68.8±1.0	94.7±0.4	81.5±32.0	41.4±4.2	53.0±9.0				
ROS	98.7±0.5	99.3±0.4	93.0±2.5	96.0±1.5	99.9±0.0	100.0±0.0	99.4±0.0	**99.7±0.0**				
	D → A				W → A				Avg.			
	OS	OS*	UNK	**HOS**	OS	OS*	UNK	**HOS**	OS	OS*	UNK	**HOS**
STA_{sum}	90.5±1.9	94.1±2.1	55.0±1.7	69.4±1.2	87.9±7.6	92.1±9.0	46.2±8.2	60.9±5.4	90.6±1.8	94.6±2.0	50.5±0.8	65.5±0.3
STA_{max}	81.5±5.1	83.1±6.2	65.9±5.0	73.2±0.9	66.4±14.5	66.2±16.3	68.0±5.2	66.1±7.1	82.6±2.1	84.3±2.4	64.8±0.9	72.5±0.8
OSBP	75.8±0.3	76.1±0.4	72.3±1.2	75.1±1.2	73.1±0.2	73.0±0.2	74.4±0.7	73.7±0.3	86.6±0.1	87.2±0.1	80.4±0.7	83.7±0.4
UAN	89.9±0.2	93.5±0.1	53.4±0.6	68.0±0.5	89.5±0.6	94.1±0.2	38.8±0.5	54.9±0.5	91.4±0.1	93.4±5.3	40.3±0.7	55.1±1.4
ROS	75.4±0.8	74.8±1.0	81.2±0.9	**77.9±0.2**	71.3±0.5	69.7±0.6	86.6±2.8	**77.2±1.0**	86.6±0.4	86.6±0.5	85.8±0.1	**85.9±0.2**
Office-31 VGGNet												
	A → W				A → D							
	OS	OS*	UNK	**HOS**	OS	OS*	UNK	**HOS**				
OSBP [12]	79.1±0.8	79.4±1.2	75.8±3.4	77.5±1.2	86.8±6.3	87.9±6.4	75.2±6.1	**81.0±6.0**				
ROS	80.4±2.4	80.3±2.5	81.7±1.7	**81.0±2.1**	81.3±1.0	81.8±1.0	76.5±0.6	79.0±0.8				
AoD [4]	86.4	87.7	73.4	79.9	90.1	92.0	71.1	79.3				
	D → W				W → D							
	OS	OS*	UNK	**HOS**	OS	OS*	UNK	**HOS**				
OSBP	96.4±0.6	96.8±0.7	93.4±0.8	**95.0±0.4**	97.5±0.3	98.9±0.4	84.2±0.8	91.0±0.5				
ROS	98.7±0.4	99.5±0.4	89.9±0.9	94.4±0.6	99.4±0.0	99.3±0.0	100.0±0.0	**99.7±0.0**				
AoD [4]	97.9	99.8	78.9	88.1	98.2	99.3	87.2	92.9				
	D → A				W → A				Avg.			
	OS	OS*	UNK	**HOS**	OS	OS*	UNK	**HOS**	OS	OS*	UNK	**HOS**
OSBP	75.1±1.3	74.4±1.5	82.4±1.2	**78.2±0.6**	70.3±4.5	69.7±4.5	76.4±5.3	72.9±4.8	84.2±0.4	84.5±0.4	81.2±1.4	82.6±0.8
ROS	77.0±0.0	76.7±0.4	79.6±3.5	78.1±1.4	64.9±0.2	62.2±0.2	91.6±0.5	**74.1±0.0**	83.6±0.4	83.3±0.4	86.5±0.3	**84.4±0.2**
AoD [4]	81.6	88.4	13.6	23.6	80.3	82.6	57.3	67.7	89.1	91.6	63.6	71.9

loss weights are set as $\lambda_{1,1} = \lambda_{2,2} = 3$ and $\lambda_{1,2} = \lambda_{2,1} = 0.1$. We ran ROS with 80 epochs for Stage I and 80 for Stage II. Each experiment is repeated three times taking the result on the target at the last epoch.

Office-31, VGGNet : batch size 32, learning rate defined as specified above and decreasing during training with inverse decay scheduling. We used SGD with momentum, setting the weight decay as 0.0005 and momentum as 0.9. The loss weights are set as $\lambda_{1,1} = \lambda_{2,2} = 3$ and $\lambda_{1,2} = \lambda_{2,1} = 0.1$. We ran ROS with 100 epochs for Stage I and 200 for Stage II. Each experiment is repeated three times taking the result on the target at the last epoch.

Office-Home, ResNet-50 : batch size 32, learning rate defined as specified above and decreasing during training with inverse decay scheduling. We used SGD with momentum, setting the weight decay as 0.0005 and momentum as 0.9. The loss weights are set as $\lambda_{1,1} = \lambda_{2,2} = 3$ and $\lambda_{2,1} = 0.1$. With respect to the previous cases, for this dataset adding the center loss to the rotation classifier R_1 seems less relevant: we kept it in the optimization process with a low weight $\lambda_{1,2} = 0.001$. We ran ROS with 150 epochs for Stage I and 45 for Stage II. Each experiment is repeated three times taking the result on the target at the last epoch.

Table 3s. Reported vs reproduced OS accuracy (%) averaged over three runs on all the sub-domains of Office-31 and Office-Home with the indicated backbones.

Reproducibility Study												
Office-31 (ResNet-50)						Office-31 (VGGNet)			Office-Home (ResNet-50)			
STA_{sum}			UAN			OSBP			STA_{sum}			UAN		
$OS_{reported}$	OS_{ours}	gap	$OS_{reported}$	OS_{ours}	gap	$OS_{reported}$	OS_{ours}	gap	$OS_{reported}$	OS_{ours}	gap	$OS_{reported}$	OS_{ours}	gap
92.9	90.6±1.8	**2.3**	89.2	87.9±0.03	**1.3**	89.1	84.2 ±0.4	**4.9**	69.5	63.3±2.1	**6.2**	77.0	75.1 ±0.2	**1.9**

It is worth noting that we essentially use the same set of parameters for all settings. This highlights that our method can generalize across datasets and network architectures without specific finetuning of the hyper-parameters. For the sake of completeness, we also provide a fully detailed evaluation of ROS including the OS metric and standard deviation for all our experiments in Tables 1s and 2s. We remark that, in terms of OS and OS*, STA is extremely unstable with large standard deviations over multiple runs.

2 Reproducibility Study

We extend here the reproducibility study presented in the main paper considering also further results on the Office-Home dataset. Specifically, in Table 3s we compare the results published in the official papers of STA [9], OSBP [12], and UAN [17] considering the OS accuracy since it is the only metric shared by all the works. For UAN we replicated the particular settings described in the original publication: for Office-Home the first 10 classes in alphabetic order are shared between source and target, the next five are private source classes and all the others are private target classes. For Office-31 the first 10 classes in alphabetic order are shared between source and target, the next 10 are private source classes and all the others are private target classes. It is worth noting that, although we used the code provided by the authors and we followed the instructions provided in the related papers, the obtained results are lower than the declared ones, with gaps that range between 1.9% and 6.2%. For complete transparency, we summarize here all the details about implementation, code and hyper-parameters used for running the competitor methods.

STA [9] `https://github.com/thuml/Separate_to_Adapt`
The code provides a full description of how to run STA for the the A→D domain shift of Office-31 with ResNet-50 backbone. For the experiments on Office-31 we trained for 900 iterations in Stage I (400 for the multi-binary classifier and 500 for the known/unknown classifier) and 1900 iterations in Stage II. We used batch size 32, SGD with momentum 0.9 and weight decay 0.0005. We used the inverse scheduling for the learning rate that is set as 0.001 in Stage I and 0.0005 in Stage II (10 times smaller for finetuned layers). Since the paper does not differentiate between Office-31 and Office-Home in terms of hyper-parameters, we ran the experiments on Office-Home with the same exact values.

It is worth noting that there is some ambiguity around the value of the learning rate for STA. The paper indicates that the learning rate may be adjusted

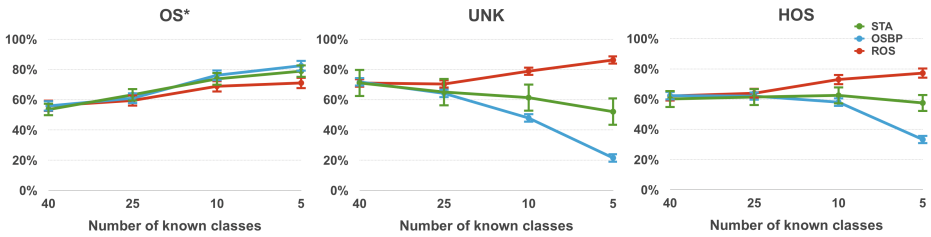
in the $\{0.001, 1\}$ range with cross-validation. However the code does not provide any validation routine, thus it is unclear how this parameter should be further refined. In addition, the learning rate set for Stage II in the code is outside the range indicated in the paper. In our experiments, we kept always the learning rate set in the code. We also found other ambiguities between the paper and the code. The paper indicates that in Stage I the feature extractor is trained on the source samples, while the feature extractor is frozen with the original weights from ImageNet. Moreover, as already discussed in our main submission, the paper presents a similarity score based on the *max* operator, while it is implemented with a *sum* operator in the released code. Finally, although the paper includes results with VGGNet, the code for this variant is not provided, nor specific details are discussed in the paper, which prevents reproducibility.

OSBP [12] `https://github.com/ksaito-ut/OPDA_BP`
This repository provides the code for OSBP, both with the VGGNet and ResNet-50 backbones. Specifically, the instructions explain how to run OSBP on the VisDA-2017 dataset [10] with VGG-19. For the experiments using VGGNet on Office-31, we used the provided implementation and we followed the description of the OSDA paper in using batch size 32, SGD with momentum 0.9, learning rate 0.001, and weight decay 0.0005. We trained only the new layers for 500 epochs, while the others were frozen with ImageNet weights. For the experiments using ResNet-50, we use batch size 32, learning rate 0.001, and train for 300 epochs for Office-Home and 500 for Office-31. Since the authors mentioned that the library version can make a significant difference in the results, for all the experiments we used exactly their declared version (Pytorch 0.3).

UAN [17] `https://github.com/thuml/Universal-Domain-Adaptation`
This repository provides the code for UAN with ResNet-50 as backbone: specific files contain instructions to run experiments on both Office-31 and Office-Home. On Office-31 we trained for 20000 iterations with batch size 36, SGD with and momentum 0.9, learning rate 0.001 for new layers and 0.0001 for finetuned layers with inverse scheduling, and weight decay 0.0005. On Office-Home we trained for 40000 iterations with batch size 36, SGD with and momentum 0.9, learning rate 0.01 for new layers and 0.001 for finetuned layers with inverse scheduling, and weight decay 0.0005. It is worth noting that the original evaluation implemented in the code would have saved the performance of UAN on the test data at each epoch and presented the best accuracy (OS) at the end of the training. This is not a standard procedure. To avoid its possibly unfair beneficial effect we provide the results obtained after the last epoch as done for all the other benchmark methods in our experiments.

3 Extended Openness Analysis

Following the openness analysis of Figure 4 of the main paper, we also extend the evaluation to include a case with lower openness: **40** known classes $\mathbb{O} =$



Office-Home (Avg.)								
	40 known classes				25 known classes			
	OS	OS*	UNK	**HOS**	OS	OS*	UNK	**HOS**
STA_{sum}[9]	53.9±1.7	53.5±1.7	71.1±3.0	60.1±1.7	63.5±2.6	63.3±3.3	65.1±2.2	61.4±1.5
OSBP[12]	56.3±1.6	55.9±1.6	71.8±3.6	**62.3±1.8**	61.4±2.5	61.0±2.8	64.2±2.8	62.0±2.8
ROS	56.2±1.2	55.9±1.2	71.1±2.8	62.1±1.8	59.9±1.9	59.5±1.9	70.4±4.0	**63.9±2.7**
	10 known classes				5 known classes			
	OS	OS*	UNK	**HOS**	OS	OS*	UNK	**HOS**
STA_{sum}[9]	72.5±5.8	73.8±6.6	61.4±14.5	62.5±8.6	74.4±2.8	78.9±6.3	52.1±17.7	57.5±12.3
OSBP[12]	73.6±5.7	76.2±5.7	47.9±4.8	58.0±5.1	72.3±4.9	82.5±5.6	21.5±1.6	33.3±2.6
ROS	69.9±5.2	68.9±5.5	78.9±2.9	**73.0±4.4**	73.4±6.7	71.1±7.6	86.3±2.7	**77.2±5.9**

Fig. 1s. Accuracy (%) averaged over the three configurations designed for each degree of openness considered: with 40, 25, 10 and 5 known classes. The table reports in details the values used to prepare the plots

$1 - \frac{40}{65} = 0.38$ using ID:{0-39, 15-54, 25-64}. The results in Figure 1s confirms the trend already observed in the main paper. Given the low UNK and HOS results of UAN we did not include this method in the ablation and focused only on the two best competitors of ROS: OSBP and STA.

4 Sensitivity analysis of the hyper-parameters

We perform a sensitivity analysis to evaluate the impact of changes in the hyper-parameter values on the performance of ROS. The experiments are performed on Office-31 with ResNet-50 as backbone and the results are displayed in Figure 6s. ROS is not very sensitive to the value of the hyper-parameters, with only $\lambda_{2,1}$ causing a variation in HOS > 1.0. Please note that it is safe to set the entropy weight to 0.1 without hyper-parameters tuning, exactly as done in [2, 9, 16]. Regardless of the specific hyper-parameters, ROS outperforms its best competitor OSBP (HOS=83.7) confirming that the superior performance is the result of algorithmic novelty rather than from hyper-parameters tuning. Moreover, we underline that we use the same hyper-parameters for all 18 domain pairs demonstrating that the choice of the hyper-parameters' value is robust across datasets. As a final remark, we note that ROS has a comparable number of parameters with respect to competing approaches. Indeed $\lambda_{1,1}$ and $\lambda_{2,2}$ are defined separaterly, but they are in fact constrained to the same value. So overall ROS has three parameters and two for the training iterations, the same as the most recent AoD (see Equation (3) of [4]).

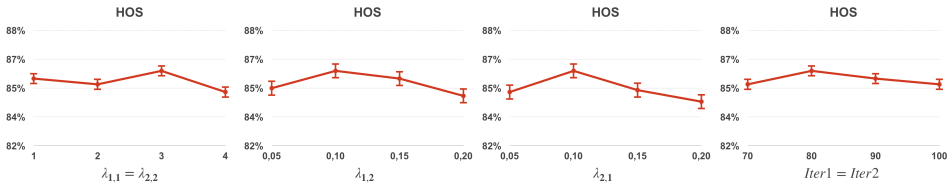


Fig. 6s. Hyper-parameter analysis

Table 4s. Analysis on the use of self-supervised tasks for the two stages of the method and further ablation.

Other Self-Supervised Tasks & Ablation Study							
STAGE I (AUC-ROC)	A → W	A → D	D → W	W → D	D → A	W → A	**Avg.**
ROS	90.1	88.1	99.4	99.9	87.5	83.8	**91.5**
ROS - Translation	80.8	74.9	82.2	98.8	72.0	79.1	81.3
ROS - Rotation+Translation	82.4	79.3	99.0	99.4	82.6	82.8	87.6
ROS - 4-Class Rotation	58.7	57.2	70.0	78.4	55.8	56.9	62.9
STAGE II (HOS)	A → W	A → D	D → W	W → D	D → A	W → A	**Avg.**
ROS	82.1	82.4	96.0	99.7	77.9	77.2	**85.9**
ROS - Jigsaw	83.1	79.3	93.5	100.0	75.5	76.1	84.6
ROS - Rotation+Jigsaw	85.7	80.5	95.0	100.0	76.0	76.7	85.7
ROS Stage I - $\lambda_{2,1} = 0$ Stage II	79.4	82.0	95.3	99.6	75.1	72.5	84.0
ROS Stage I - ROS Stage II+Center Loss	79.6	82.8	95.1	99.5	77.8	76.3	85.2

Table 5s. Runtime analysis on Office-31(A-W) with ResNet-50. Hardware - CPU: Intel(R) Core(TM) i7-5930K @ 3.50GHz, GPU (x1): Nvidia GeForce GTX 1080Ti.

Time analysis			
STA[9]	UAN[17]	OSBP[12]	**ROS**
1069s	9615s	3672s	1875s

5 Other Self-Supervised Tasks and Further Ablation

Our goal is to show that it is possible to successfully tackle both sub-tasks of OSDA, known/unknown separation and domain alignment, with a single self-supervised task. From the literature of CSDA [15, 2] and anomaly detection [5, 1, 7], rotation classification clearly emerges as the most reliable candidate for our purpose. To confirm our claim, we run additional experiments on Office-31 (ResNet-50) with alternative self-supervised tasks. Following [5], we considered the self-supervised task of translation classification for anomaly detection (Stage I). Moreover, following [2], we considered the self-supervised task of solving a jigsaw puzzle for domain alignment (Stage II). Table 4s show the obtained results: in both sets of experiments, rotation recognition alone outperforms both the alternative task and combination of the two tasks.

We also confirm the crucial contribution of the multi-rotation task instead of the standard 4-Class task in Stage I. Table 4s shows that the standard rotation decreases the AUC-ROC by an astonishing 28.6%. Of course we keep the anchor (relative rotation) also in this 4-Class experiment.

Since using the entropy loss in the object classification process across domains is standard practice, we did not include an ablation for Stage II of ROS on this term in the main paper. For completeness we present it here. We set $\lambda_{2,1} = 0$ including the results in Table 4s: as expected, without the entropy loss the performance drop on average of 1.9 percentage points, confirming that the entropy helps to adapt with a more evident effect in case of large domain gaps (*e.g.* A→W, W→A). Moreover, in Stage II, the center loss is not as relevant as for Stage I, and it would imply the introduction of an extra hyper-parameter.

Algorithm 1 Compute normality score and Generate $\mathcal{D}_t^{knw}$ & $\mathcal{D}_t^{unk}$

Input:
Trained networks E and R_1
Target dataset $\mathcal{D}_t = \{\boldsymbol{x}_j^t\}_{j=1}^{N_t}$
Output:
Known target dataset $\mathcal{D}_t^{knw} = \{\boldsymbol{x}_j^{t,knw}\}_{j=1}^{N_{t,knw}}$
Unknown target dataset $\mathcal{D}_t^{unk} = \{\boldsymbol{x}_j^{t,unk}\}_{j=1}^{N_{t,unk}}$

procedure getRotationScore($\boldsymbol{z}$,i)
  $\mathbf{o} = zeros(|\mathcal{C}_s|)$ # vector of $|\mathcal{C}_s|$ zeros
  for each k **in** $\{1, ..., |\mathcal{C}_s|\}$ **do**
    $[\mathbf{o}]_k = [\boldsymbol{z}]_{k\times 4+i}$ # $[\mathbf{a}]_b$ indicated the b-th element of vector $\mathbf{a}$
  return o
procedure getEntropyScore($\boldsymbol{z}$)
  return $\boldsymbol{z} \cdot \log(\boldsymbol{z}) / \log(|\mathcal{C}_s|)$
procedure getNormalityScore(E,R_1,$\mathcal{D}_t$)
  for each $\boldsymbol{x}_j^t$ **in** $\mathcal{D}_t$ **do**
    Initialize: $h = \{\}$, $\mathbf{o} = zeros(|\mathcal{C}_s|)$
    for each i **in** $\{1, ..., 4\}$ **do**
      $\tilde{\boldsymbol{x}}_j = rot90(\boldsymbol{x}_j, i)$
      $\boldsymbol{z}_j = \text{softmax}\big(R_1(E(\boldsymbol{x}_j)||E(\tilde{\boldsymbol{x}}_j))\big)$
      $h \leftarrow getEntropyScore(\boldsymbol{z}_j)$
      $o \mathrel{+}= getRotationScore(\boldsymbol{z}_j, i)$ # element-wise sum of vectors
    $h = mean(h)$
    $o = max(\mathbf{o})$
    $\mathcal{N} \leftarrow \eta_j = max(o, 1 - h)$
  return $\mathcal{N}$

procedure Main( )
  Initialize: $\mathcal{D}_t^{knw} = \{\}$, $\mathcal{D}_t^{unk} = \{\}$
  $\mathcal{A} = getNormalityScore(E, R_1, \mathcal{D}_t)$
  for each $(\boldsymbol{x}_j, \eta_j)$ **in** $(\mathcal{D}_t, \mathcal{N})$ **do**
    if $\eta_j \geq mean(\mathcal{N})$ **then**
      $\mathcal{D}_t^{knw} \leftarrow \boldsymbol{x}_j$
    else
      $\mathcal{D}_t^{unk} \leftarrow \boldsymbol{x}_j$

Indeed, the results in in Table 4s indicate that adding the center loss to Stage II might even produce a slight drop in performance.

6 Time analysis

We executed a training runtime analysis on Office-31(A-W) with ResNet-50 for all the methods discussed in the paper with their indicated hyper-parameters. The results in Table 5s show that that the time is not an issue and ROS is even twice as fast as its best competitor in terms of HOS performance (OSBP).

7 Normality Score Pseudo-code

As promised in the main paper we summarize in Algorithm 1 the procedure used to calculate the normality score at the end of Stage I of ROS.

References

1. Bergman, L., Hoshen, Y.: Classification-based anomaly detection for general data. In: ICLR (2020)
2. Carlucci, F.M., D'Innocente, A., Bucci, S., Caputo, B., Tommasi, T.: Domain generalization by solving jigsaw puzzles. In: CVPR (2019)
3. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: CVPR (2009)
4. Feng, Q., Kang, G., Fan, H., Yang, Y.: Attract or distract: Exploit the margin of open set. In: CVPR (2019)
5. Golan, I., El-Yaniv, R.: Deep anomaly detection using geometric transformations. In: NeurIPS. pp. 9758–9769 (2018)
6. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016)
7. Hendrycks, D., Mazeika, M., Kadavath, S., Song, D.: Using self-supervised learning can improve model robustness and uncertainty. In: NeurIPS (2019)
8. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. Preprint arXiv:1502.03167 (2015)
9. Liu, H., Cao, Z., Long, M., Wang, J., Yang, Q.: Separate to adapt: Open set domain adaptation via progressive separation. In: CVPR (2019)
10. Peng, X., Usman, B., Kaushik, N., Hoffman, J., Wang, D., Saenko, K.: Visda: The visual domain adaptation challenge. Preprint arXiv:1710.06924 (2017)
11. Saenko, K., Kulis, B., Fritz, M., Darrell, T.: Adapting visual category models to new domains. In: ECCV (2010)
12. Saito, K., Yamamoto, S., Ushiku, Y., Harada, T.: Open set domain adaptation by backpropagation. In: ECCV (2018)
13. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. Preprint arXiv:1409.1556 (2014)
14. Venkateswara, H., Eusebio, J., Chakraborty, S., Panchanathan, S.: Deep hashing network for unsupervised domain adaptation. In: CVPR (2017)
15. Xu, J., Xiao, L., López, A.M.: Self-supervised domain adaptation for computer vision tasks. IEEE Access **7**, 156694–156706 (2019)

16. Xu, R., Li, G., Yang, J., Lin, L.: Larger norm more transferable: An adaptive feature norm approach for unsupervised domain adaptation. In: ICCV (2019)
17. You, K., Long, M., Cao, Z., Wang, J., Jordan, M.I.: Universal domain adaptation. In: CVPR (2019)