# On the Effectiveness of Image Rotation for Open Set Domain Adaptation

Silvia Bucci*,1,2[0000−0001−6318−7288], Mohammad Reza Loghmani*,3[0000−0002−2687−7877], and Tatiana Tommasi1,2[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

**Abstract.** Open Set Domain Adaptation (OSDA) bridges the domain gap between a labeled source domain and an unlabeled target domain, while also rejecting target classes that are not present in the source. To avoid negative transfer, OSDA can be tackled by first separating the known/unknown target samples and then aligning known target samples with the source data. We propose a novel method to addresses both these problems using the self-supervised task of rotation recognition. Moreover, we assess the performance with a new open set metric that properly balances the contribution of recognizing the known classes and rejecting the unknown samples. Comparative experiments with existing OSDA methods on the standard Office-31 and Office-Home benchmarks show that: (i) our method outperforms its competitors, (ii) reproducibility for this field is a crucial issue to tackle, (iii) our metric provides a reliable tool to allow fair open set evaluation.

**Keywords:** Open Set Domain Adaptation · Self-supervised Learning

## 1 Introduction

The current success of deep learning models is showing how modern artificial intelligent systems can manage supervised machine learning tasks with growing accuracy. However, when the level of supervision decreases, all the limitations of the existing data-hungry approaches become evident. For many applications, large amount of supervised data are not readily available, moreover collecting and manually annotating such data may be difficult or very costly. Different sub-fields of computer vision, such as *domain adaptation* [8] and *self-supervised learning* [11], aim at designing new learning solutions to compensate for this lack of supervision. Domain adaptation focuses on leveraging a fully supervised data-rich source domain to learn a classification model that performs well on a different but related unlabeled target domain. Traditional domain adaptation

* equal contributions

methods assume that the target contains exactly the same set of labels of the source (*closed-set* scenario). In recent years, this constraint has been relaxed in favor of the more realistic *open-set* scenario where the target also contains samples drawn from unknown classes. In this case, it becomes important to identify and isolate the unknown class samples before reducing the domain shift to avoid negative transfer. Self-supervised learning focuses on training models on pretext tasks, such as image colorization or rotation prediction, using unlabeled data to then transfer the acquired high-level knowledge to new tasks with scarce supervision. Recent literature has highlighted how self-supervision can be used for domain adaptation: jointly solving a pretext self-supervised task together with the main supervised problem leads to learning robust cross-domain features and supports generalization [47,5]. Other works have also shown that the output of self-supervised models can be used in anomaly detection to discriminate normal and anomalous data [17,2]. However, these works only tackle binary problems (normal and anomalous class) and deal with a single domain.

In this paper, we propose for the first time to use the inherent properties of self-supervision both for cross-domain robustness and for novelty detection to solve *Open-Set Domain Adaptation* (OSDA). To this purpose, we propose a two-stage method called *Rotation-based Open Set* (ROS) that is illustrated in Figure 1. In the first stage, we separate the known and unknown target samples by training the model on a modified version of the rotation task that consists in predicting the relative rotation between a reference image and the rotated counterpart. In the second stage, we reduce the domain shift between the source domain and the known target domain using, once again, the rotation task. Finally we obtain a classifier that predicts each target sample as either belonging to one of the known classes or rejects it as unknown. While evaluating ROS on the two popular benchmarks *Office-31* [36] and *Office-Home* [44], we expose the reproducibility problem of existing OSDA approaches and assess them with a new evaluation metric that better represents the performance of open set methods. **We can summarize the contributions of our work as following**:

1. we introduce a novel OSDA method that exploits rotation recognition to tackle both known/unknown target separation and domain alignment;
2. we define a new OSDA metric that properly accounts for both known class recognition and unknown rejection;
3. we present an extensive experimental benchmark against existing OSDA methods with two conclusions: (a) we put under the spotlight the urgent need of a rigorous experimental validation to guarantee result reproducibility; (b) our ROS defines the new state-of-the-art on two benchmark datasets.

A Pytorch implementation of our method, together with instructions to replicate our experiments, is available at `https://github.com/silvia1993/ROS` .

## 2 Related Work

**Self-supervised learning** applies the techniques of supervised learning on problems where external supervision is not available. The idea is to manipu-

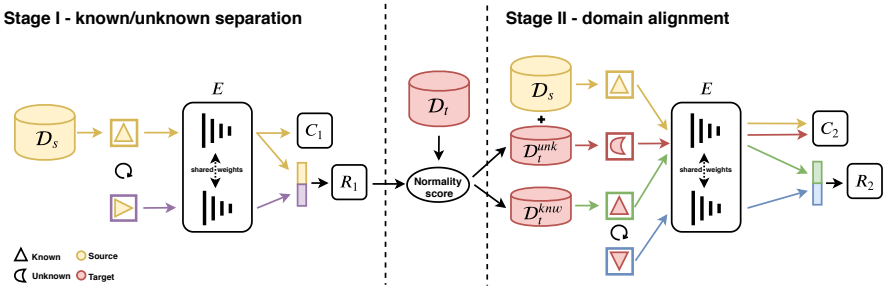



**Fig. 1.** Schematic illustration of our Rotation-based Open Set (**ROS**). Stage I: the source dataset $\mathcal{D}_s$ is used to train the encoder $E$, the semantic classifier $C_1$, and the multi-rotation classifier $R_1$ to perform known/unknown separation. $C_1$ is trained using the features of the original image, while $R_1$ is trained using the concatenated features of the original and rotated image. After convergence, the prediction of $R_1$ on the target dataset $\mathcal{D}_t$ is used to generate a normality score that defines how the target samples are split into a known target dataset $\mathcal{D}_t^{knw}$ and an unknown target dataset $\mathcal{D}_t^{unk}$. Stage II: $E$, the semantic+unknown classifier $C_2$ and the rotation classifier $R_2$ are trained to align the source and target distributions and to recognize the known classes while rejecting the unknowns. $C_2$ is trained using the original images from $\mathcal{D}_s$ and $\mathcal{D}_t^{unk}$, while $R_2$ is trained using the concatenated features of the original and rotated known target samples.

late the data to generate the supervision for an artificial task that is helpful to learn useful feature representations. Examples of self-supervised tasks in computer vision include predicting the relative position of image patches [11,30], colorizing a gray-scale image [51,24], and inpainting a removed patch [32]. Arguably, one of the most effective self-supervised tasks is rotation recognition [16] that consists in rotating the input images by multiples of 90° and training the network to predict the rotation angle of each image. This pretext task has been successfully used in a variety of applications including anomaly detection [17] and closed-set domain adaptation [47].

**Anomaly detection**, also known as outlier or novelty detection, aims at learning a model from a set of *normal* samples to be able to detect out-of-distribution (*anomalous*) instances. The research literature in this area is wide with three main kind of approaches. *Distribution-based* methods [53,23,50,54] model the distribution of the available normal data so that the anomalous samples can be recognized as those with a low likelihood under the learned probability function. *Reconstruction-based* methods [12,4,46,52,39] learn to reconstruct the normal samples from an embedding or a set of basis functions. Anomalous data are then recognized by having a larger reconstruction error with respect to normal samples. *Discriminative* methods [40,34,20,25] train a classifier on the normal data and use its predictions to distinguish between normal and anomalous samples.

**Closed-set domain adaptation** (CSDA) accounts for the difference between source and target data by considering them as drawn from two different

marginal distributions. The literature of DA can be divided into three groups based on the strategy used to reduce the domain shift. *Discrepancy-based* methods [28,42,48] define a metric to measure the distance between source and target data in feature space. This metric is minimized while training the network to reduce the domain shift. *Adversarial* methods [14,43,35] aim at training a domain discriminator and a generator network in an adversarial fashion so that the generator converges to a solution that makes the source and target data indistinguishable for the domain discriminator. *Self-supervised* methods [15,3,5] train a network to solve an auxiliary self-supervised task on the target (and source) data, in addition to the main task, to learn robust cross-domain representations.

**Open Set Domain Adaptation** (OSDA) is a more realistic version of CSDA, where the source and target distribution do not contain the same categories. The term "OSDA" was first introduced by Busto and Gall [31] that considered the setting where each domain contains, in addition to the shared categories, a set of private categories. The currently accepted definition of OSDA was introduced by Saito *et al*. [37] that considered the target as containing all the source categories and additional set of private categories that should be considered *unknown*. To date, only a handful of papers tackled this problem. *Open Set Back-Propagation* (OSBP) [37] is an adversarial method that consists in training a classifier to obtain a large boundary between source and target samples whereas the feature generator is trained to make the target samples far from the boundary. *Separate To Adapt* (STA) [26] is an approach based on two stages. First, a multi-binary classifier trained on the source is used to estimate the similarity of target samples to the source. Then, target data with extreme high and low similarity are re-used to separate known and unknown classes while the features across domains are aligned through adversarial adaptation. *Attract or Distract* (AoD) [13] starts with a mild alignment with a procedure similar to [37] and refines the decision by using metric learning to reduce the intra-class distance in known classes and push the unknown class away from the known classes. *Universal Adaptation Network* (UAN)[4] [49] uses a pair of domain discriminators to both generate a sample-level transferability weight and to promote the adaptation in the automatically discovered common label set. Differently from all existing OSDA methods, **our approach abandons adversarial training in favor of self-supervision. Indeed, we show that rotation recognition can be used, with tailored adjustments, both to separate known and unknown target samples and to align the known source and target distributions**[5].

[4] UAN is originally proposed for the universal domain adaptation setting that is a superset of OSDA, so it can also be used in the context of this paper.

[5] See Appendix E for a discussion on the use of other self-supervised tasks.

# 3 Method

## 3.1 Problem formulation

Let us denote with $\mathcal{D}_s = \{(\boldsymbol{x}_j^s, y_j^s)\}_{j=1}^{N_s} \sim p_s$ the labeled source dataset drawn from distribution $p_s$ and $\mathcal{D}_t = \{\boldsymbol{x}_j^t\}_{j=1}^{N_t} \sim p_t$ the unlabeled target dataset drawn from distribution $p_t$. In OSDA, the source domain is associated with a set of *known* classes $y^s \in \{1, \dots, |\mathcal{C}_s|\}$ that are shared with the target domain $\mathcal{C}_s \subset \mathcal{C}_t$, but the target covers also a set $\mathcal{C}_{t \setminus s}$ of additional classes, which are considered *unknown*. As in CSDA, it holds that $p_s \neq p_t$ and we further have that $p_s \neq p_t^{\mathcal{C}_s}$, where $p_t^{\mathcal{C}_s}$ denotes the distribution of the target domain belonging to the shared label space $\mathcal{C}_s$. Therefore, in OSDA we face both a domain gap ($p_s \neq p_t^{\mathcal{C}_s}$) and a category gap ($\mathcal{C}_s \neq \mathcal{C}_t$). OSDA approaches aim at assigning the target samples to either one of the $|\mathcal{C}_s|$ shared classes or to reject them as *unknown* using only annotated source samples, with the unlabeled target samples available transductively. An important measure characterizing a given OSDA problem is the *openness* that relates the size of the source and target class set. For a dataset pair $(\mathcal{D}_s, \mathcal{D}_t)$, following the definition of [1], the openness $\mathbb{O}$ is measured as $\mathbb{O} = 1 - \frac{|\mathcal{C}_s|}{|\mathcal{C}_t|}$. In CSDA $\mathbb{O} = 0$, while in OSDA $\mathbb{O} > 0$.

## 3.2 Overview

When designing a method for OSDA, we face two main challenges: *negative transfer* and *known/unknown separation*. Negative transfer occurs when the whole source and target distribution are forcefully matched, thus also the unknown target samples are mistakenly aligned with source data. To avoid this issue, cross-domain adaptation should focus only on the shared $\mathcal{C}_s$ classes, closing the gap between $p_t^{\mathcal{C}_s}$ and $p_s$. This leads to the challenge of known/unknown separation: recognizing each target sample as either belonging to one of the shared classes $\mathcal{C}_s$ (known) or to one of the target private classes $\mathcal{C}_{t \setminus s}$ (unknown). Following these observations, we structure our approach in two stages: (i) we separate the target samples into known and unknown, and (ii) we align the target samples predicted as known with the source samples (see Figure 1). The first stage is formulated as an anomaly detection problem where the unknown samples are considered as anomalies. The second stage is formulated as a CSDA problem between source and the known target distribution. Inspired by recent advances in anomaly detection and CSDA [47,17], we solve both stages using the power of self-supervision. More specifically, we use two variations of the rotation classification task to compute a normality score for the known/unknown separation of the target samples and to reduce the domain gap.

## 3.3 Rotation recognition for open set domain adaptation

Let us denote with $rot90(\boldsymbol{x}, i)$ the function that rotates clockwise a 2D image $\boldsymbol{x}$ by $i \times 90°$. Rotation recognition is a self-supervised task that consists in rotating

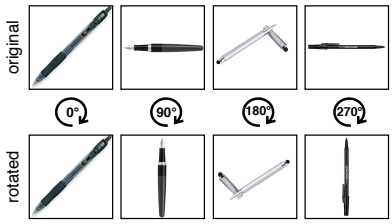



**Fig. 2.** Are you able to infer the rotation degree of the rotated images without looking at the respective original one?

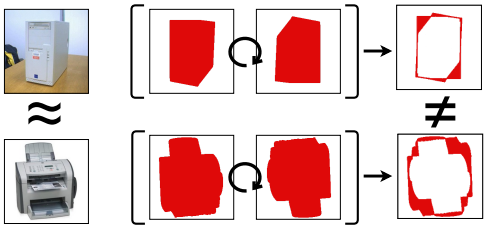


**Fig. 3.** The objects on the left may be confused. The relative rotation guides the network to focus on discriminative shape information

a given image $x$ by a random $i \in [1, 4]$ and using a CNN to predict $i$ from the rotated image $\tilde{\boldsymbol{x}} = rot90(\boldsymbol{x}, i)$. We indicate with $|r| = 4$ the cardinality of the label space for this classification task. In order to effectively apply rotation recognition to OSDA, we introduce the following variations.

*Relative rotation:* Consider the images in Figure 2. Inferring by how much each image has been rotated without looking at its original (non-rotated) version is an ill-posed problem since the pens, as all the other object classes, are not presented with a coherent orientation in the dataset. On the other hand, looking at both original and rotated image to infer the relative rotation between them is well-defined. Following this logic, we modify the standard rotation classification task [16] by introducing the original image as an anchor. Finally, we train the rotation classifier to predict the rotation angle given the concatenated features of both original (anchor) and rotated image. As indicated by Figure 3, the proposed relative rotation has the further effect of boosting the discriminative power of the learned features. It guides the network to focus more on specific shape details rather than on confusing texture information across different object classes.

*Multi-rotation classification:* The standard setting of anomaly detection considers samples from one semantic category as the normal class and samples from other semantic categories as anomalies. Rotation recognition has been successfully applied to this setting, but it suffers when including multiple semantic categories in the normal class [17]. This is the case when coping with the known/unknown separation of OSDA, where we have all the $|\mathcal{C}_s|$ semantic categories as known data. To overcome this problem, we propose a simple solution: we extend rotation recognition from a 4-class problem to a $(4 \times |\mathcal{C}_s|)$-class problem, where the set of classes represents the combination of semantic and rotation labels. For example, if we rotate an image of category $y^s = 2$ by $i = 3$, its label for the multi-rotation classification task is $z^s = (y^s \times 4) + i = 11$. In Appendix E, we discuss the specific merits of the multi-rotation classification task with further experimental evidences. In the following, we indicate with $\boldsymbol{y}, \boldsymbol{z}$ the one-hot vectors respectively for the class and multi-rotation labels.

### 3.4 Stage I: known/unknown separation

To distinguish between the known and unknown samples of $\mathcal{D}_t$, we train a CNN on the multi-rotation classification task using $\tilde{\mathcal{D}}_s = \{(\boldsymbol{x}_j^s, \tilde{\boldsymbol{x}}_j^s, z_j^s)\}_{j=1}^{4\times N_s}$. The network is composed of an encoder $E$ and two heads: a multi-rotation classifier $R_1$ and a semantic label classifier $C_1$. The rotation prediction is computed on the stacked features of the original and rotated image produced by the encoder $\hat{\boldsymbol{z}}^s = \text{softmax}\big(R_1([E(\boldsymbol{x}^s), E(\tilde{\boldsymbol{x}}^s)])\big)$, while the semantic prediction is computed only from the original image features as $\hat{\boldsymbol{y}}^s = \text{softmax}\big(C_1(E(\boldsymbol{x}^s)\big)$. The network is trained to minimize the objective function $\mathcal{L}_1 = \mathcal{L}_{C_1} + \mathcal{L}_{R_1}$, where the semantic loss $\mathcal{L}_{C_1}$ is defined as a cross-entropy and the multi-rotation loss $\mathcal{L}_{R_1}$ combines cross-entropy and center loss [45]. More precisely,

$$\mathcal{L}_{C_1} = -\sum_{j\in\mathcal{D}_s} \boldsymbol{y}_j^s \cdot \log(\hat{\boldsymbol{y}}_j^s), \tag{1}$$

$$\mathcal{L}_{R_1} = \sum_{j\in\tilde{\mathcal{D}}_s} -\lambda_{1,1}\boldsymbol{z}_j^s \cdot \log(\hat{\boldsymbol{z}}_j^s) + \lambda_{1,2}||\boldsymbol{v}_j^s - \gamma(\boldsymbol{z}_j^s)||_2^2, \tag{2}$$

where $||.||_2$ indicates the $l_2$-norm operator, $\boldsymbol{v}_j$ indicates the output of the penultimate layer of $R_1$ and $\gamma(\boldsymbol{z}_j)$ indicates the corresponding centroid of the class associated with $\boldsymbol{v}_j$. By using the center loss we further encourage the network to minimize the intra-class variations while keeping far the features of different classes. This supports the following use of the rotation classifier output as a metric to detect unknown category samples.

Once the training is complete, we use $E$ and $R_1$ to compute the *normality score* $\mathcal{N} \in [0,1]$ for each target sample, with large $\mathcal{N}$ values indicating normal (known) samples and vice-versa. We start from the network prediction on all the relative rotation variants of a target sample $\hat{\boldsymbol{z}}_i{}^t = \text{softmax}\big(R_1([E(\boldsymbol{x}^t), E(\tilde{\boldsymbol{x}}_i^t)])\big)_i$ and their related entropy $H(\hat{\boldsymbol{z}}_i^t) = \big(\hat{\boldsymbol{z}}_i^t \cdot \log(\hat{\boldsymbol{z}}_i^t)/\log|\mathcal{C}_s|\big)_i$ with $i = 1,\ldots,|r|$. We indicate with $[\hat{\boldsymbol{z}}^t]_m$ the $m$-th component of the $\hat{\boldsymbol{z}}^t$ vector. The full expression of the normality score is:

$$\mathcal{N}(\boldsymbol{x}^t) = \max\left\{ \max_{k=1,\ldots,|\mathcal{C}_s|} \left( \sum_{i=1}^{|r|} [\hat{\boldsymbol{z}}_i^t]_{k\times|r|+i} \right), \left(1 - \frac{1}{|r|}\sum_{i=1}^{|r|} H(\hat{\boldsymbol{z}}_i^t)\right) \right\}. \tag{3}$$

In words, this formula is a function of the ability of the network to correctly predict the semantic class and orientation of a target sample (first term in the braces, *Rotation Score*) as well as of its confidence evaluated on the basis of the prediction entropy (second term, *Entropy Score*). We maximize over these two components with the aim of taking the most reliable metric in each case. Finally, the normality score is used to separate the target dataset into a known target dataset $\mathcal{D}_t^{knw}$ and an unknown target dataset $\mathcal{D}_t^{unk}$. The distinction is made directly through the data statistics using the average of the normality score over the whole target $\bar{\mathcal{N}} = \frac{1}{N_t}\sum_{j=1}^{N_t}\mathcal{N}_j$, without the need to introduce any further parameter:

$$\begin{cases} \boldsymbol{x}^t \in \mathcal{D}_t^{knw} & \text{if} \quad \mathcal{N}(\boldsymbol{x}^t) > \bar{\mathcal{N}} \\ \boldsymbol{x}^t \in \mathcal{D}_t^{unk} & \text{if} \quad \mathcal{N}(\boldsymbol{x}^t) < \bar{\mathcal{N}} \end{cases}. \tag{4}$$

It is worth mentioning that only $R_1$ is directly involved in computing the normality score, while $C_1$ is only trained for regularization purposes and as a warm up for the following stage. For a detailed pseudo-code on how to compute $\mathcal{N}$ and generate $\mathcal{D}_t^{knw}$ and $\mathcal{D}_t^{unk}$, please refer to Appendix G.

### 3.5 Stage II: domain alignment

Once the target unknown samples have been identified, the scenario gets closer to that of standard CSDA. On the one hand, we can use $\mathcal{D}_t^{knw}$ to close the domain gap without the risk of negative transfer and, on the other hand, we can exploit $\mathcal{D}_t^{unk}$ to extend the original semantic classifier, making it able to recognize the unknown category. Similarly to Stage I, the network is composed of an encoder $E$ and two heads: a rotation classifier $R_2$ and a semantic label classifier $C_2$. The encoder is inherited from the previous stage. The heads also leverage on the previous training phase but have two key differences with respect to Stage I: (1) $C_1$ has a $|\mathcal{C}_s|$-dimensional output, while $C_2$ has a $(|\mathcal{C}_s|+1)$-dimensional output because of the addition of the unknown class; (2) $R_1$ is a multi-rotation classifier with a $(4 \times |\mathcal{C}_s|)$-dimensional output, $R_2$ is a rotation classifier with a 4-dimensional output. The rotation prediction is computed as $\hat{\boldsymbol{q}} = \text{softmax}\big(R_2([E(\boldsymbol{x}), E(\tilde{\boldsymbol{x}})])\big)$ while the semantic prediction is $\hat{\boldsymbol{g}} = \text{softmax}\big(C_2(E(\boldsymbol{x}))\big)$. The network is trained to minimize the objective function $\mathcal{L}_2 = \mathcal{L}_{C_2} + \mathcal{L}_{R_2}$, where $\mathcal{L}_{C_2}$ combines the supervised cross-entropy and the unsupervised entropy loss for the classification task, while $\mathcal{L}_{R_2}$ is defined as a cross-entropy for the rotation task. The unsupervised entropy loss is used to involve in the semantic classification process also the unlabeled target samples recognized as known. This loss enforces the decision boundary to pass through low-density areas. More precisely,

$$\mathcal{L}_{C_2} = - \sum_{j \in \{\mathcal{D}_s \cup \mathcal{D}_t^{unk}\}} \boldsymbol{g}_j \cdot \log(\hat{\boldsymbol{g}}_j) - \lambda_{2,1} \sum_{j \in \mathcal{D}_t^{knw}} \hat{\boldsymbol{g}}_j \cdot \log(\hat{\boldsymbol{g}}_j), \tag{5}$$

$$\mathcal{L}_{R_2} = -\lambda_{2,2} \sum_{j \in \mathcal{D}_t^{knw}} \boldsymbol{q}_j \cdot \log(\hat{\boldsymbol{q}}_j) \ . \tag{6}$$

Once the training is complete, $R_2$ is discarded and the target labels are simply predicted as $c_j^t = C_2(E(\boldsymbol{x}_j^t))$ for all $j = 1, \ldots, N_t$.

## 4 On reproducibility and open set metrics

OSDA is a young field of research first introduced in 2017. As it is gaining momentum, it is crucial to guarantee the *reproducibility* of the proposed methods and have a valid *metric* to properly evaluate them.

*Reproducibility:* In recent years, the machine learning community has become painfully aware of a reproducibility crisis [19,10,29]. Replicating the results of state-of-the-art deep learning models is seldom straightforward due to a combination of non-deterministic factors in standard benchmark environments and poor reports from the authors. Although the problem is far from being solved,

several efforts have been made to promote reproducibility through checklists [7], challenges [6] and by encouraging authors to submit their code. On our side, we contribute by re-running the state-of-the-art methods for OSDA and compare them with the results reported in the papers (see Section 5). Our results are produced using the original public implementation together with the parameters reported in the paper and, in some cases, repeated communications with the authors. We believe that this practice, as opposed to simply copying the results reported in the papers, can be of great value to the community.

*Open set metrics:* The usual metrics adopted to evaluate OSDA are the average class accuracy over the known classes $OS^*$, and the accuracy of the unknown class $UNK$. They are generally combined in $OS = \frac{|\mathcal{C}_s|}{|\mathcal{C}_s|+1} \times OS^* + \frac{1}{|\mathcal{C}_s|+1} \times UNK$ as a measure of the overall performance. However, we argue (and we already demonstrated in [27]) that treating the unknown as an additional class does not provide an appropriate metric. As an example, let us consider an algorithm that is not designed to deal with unknown classes ($UNK$=0.0%) but has perfect accuracy over 10 known classes ($OS^*$=100.0%). Although this algorithm is not suitable for open set scenarios because it completely disregards false positives, it presents a high score of $OS$=90.9%. With increasing number of known classes, this effect on OS becomes even more acute, making the role of $UNK$ negligible. For this reason, we propose a new metric defined as the harmonic mean of $OS^*$ and $UNK$, $HOS = 2\frac{OS^* \times UNK}{OS^* + UNK}$. Differently from $OS$, $HOS$ provides a high score only if the algorithm performs well both on known and on unknown samples, independently of $|\mathcal{C}_s|$. Using a harmonic mean instead of a simple average penalizes large gaps between $OS^*$ and $UNK$.

## 5 Experiments

### 5.1 Setup: Baselines, Datasets

We validate ROS with a thorough experimental analysis on two widely used benchmark datasets, Office-31 and Office-Home. *Office-31* [36] consists of three domains, Webcam (W), Amazon (A) and Dslr (D), each containing 31 object categories. We follow the setting proposed in [37], where the first 10 classes in alphabetic order are considered known and the last 11 classes are considered unknown. *Office-Home* [44] consists of four domains, Product (Pr), Art (Ar), Real World (Rw) and Clipart (Cl), each containing 65 object categories. Unless otherwise specified, we follow the setting proposed in [26], where the first 25 classes in alphabetic order are considered known classes and the remaining 40 classes are considered unknown. Both the number of categories and the large domain gaps make this dataset much more challenging than Office-31.

We compare ROS against the state-of-the-art methods STA [26], OSBP [37], UAN [49], AoD [13], that we already described in Section 2. For each of them, we run experiments using the official code provided by the authors, with the exact parameters declared in the relative paper. The only exception was made for AoD for which the authors have not released the code at the time of writing,

thus we report the values presented in their original work. We also highlight that STA presents a practical issue related to the similarity score used to separate known and unknown categories. Its formulation is based on the *max* operator according to the Equation (2) in [26], but appears instead based on *sum* in the implementation code. In our analysis we considered both the two variants ($\text{STA}_{\text{sum}}$, $\text{STA}_{\text{max}}$) for the sake of completeness. All the results presented in this section, both for ROS and for the baseline methods, are the average over three independent experimental runs. We do not cherry pick the best out of several trials, but only run the three experiments we report.

### 5.2 Implementation Details

By following standard practice, we evaluate the performances of ROS on Office-31 using two different backbones ResNet-50 [18] and VGGNet [41], both pre-trained on ImageNet [9], and we focus on ResNet-50 for Office-Home. The hyper-parameters values are the same regardless of the backbone and the dataset used. In particular, in both Stage I and Stage II of ROS the batch size is set to 32 with a learning rate of 0.0003 which decreases during the training following an inverse decay scheduling. For all layers trained from scratch, we set the learning rate 10 times higher than the pre-trained ones. We use SGD, setting the weight decay as 0.0005 and momentum as 0.9. In both stages, the weight of the self-supervised task is set three times the one of the semantic classification task, thus $\lambda_{1,1} = \lambda_{2,2} = 3$. In Stage I, the weight of the center loss is $\lambda_{1,2} = 0.1$ and in Stage II the weight of the entropy loss is $\lambda_{2,1} = 0.1$. The network trained in Stage I is used as starting point for Stage II. To take into consideration the extra category, 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). More implementation details and a sensitivity analysis of the hyper-parameters are provided in Appendix A and D.

### 5.3 Results

*How does our method compare to the state-of-the-art?* Table 1 and 2 show the average results over three runs on each of the domain shifts, respectively of Office-31 and Office-Home. To discuss the results, we focus on the HOS metric since it is a synthesis of OS* and UNK, as discussed in Section 4. Overall, ROS outperforms the state-of-the-art on a total of 13 out of 18 domain shifts and presents the highest average performance on both Office-31 and Office-Home. The HOS improvement gets up to 2.2% compared to the second best method OSBP. Specifically, ROS has a large gain over STA, regardless of its specific max or sum implementation, while UAN is not a challenging competitor due to its low performance on the unknown class. We can compare against AoD only when using VGG for Office-31: we report the original results in gray in Table 2, with the HOS value confirming our advantage.

A more in-depth analysis indicates that the advantage of ROS is largely related in its ability in separating known and unknown samples. Indeed, while

**Table 1.** Accuracy (%) averaged over three runs of each method on Office-31 dataset using ResNet-50 and VGGNet as backbones

| **Office-31** | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **ResNet-50** | | | | | | | | | | | | | | | | | | | | | | |
| | A → W | | | A → D | | | D → W | | | W → D | | | D → A | | | W → A | | | **Avg.** | | |
| | OS* | UNK | **HOS** | OS* | UNK | **HOS** | OS* | UNK | **HOS** | OS* | UNK | **HOS** | OS* | UNK | **HOS** | OS* | UNK | **HOS** | OS* | UNK | **HOS** |
| $STA_{sum}$ [26] | 92.1 | 58.0 | 71.0 | 95.4 | 45.5 | 61.6 | 97.1 | 49.7 | 65.5 | 96.6 | 48.5 | 64.4 | 94.1 | 55.0 | 69.4 | 92.1 | 46.2 | 60.9 | 94.6 | 50.5 | 65.5±0.3 |
| $STA_{max}$ [26] | 86.7 | 67.6 | 75.9 | 91.0 | 63.9 | 75.0 | 94.1 | 55.5 | 69.8 | 84.9 | 67.8 | 75.2 | 83.1 | 65.9 | 73.2 | 66.2 | 68.0 | 66.1 | 84.3 | 64.8 | 72.5±0.8 |
| OSBP [37] | 86.8 | 79.2 | **82.7** | 90.5 | 75.5 | **82.4** | 97.7 | 96.7 | **97.2** | 99.1 | 84.2 | 91.1 | 76.1 | 72.3 | 75.1 | 73.0 | 74.4 | 73.7 | 87.2 | 80.4 | 83.7±0.4 |
| UAN [49] | 95.5 | 31.0 | 46.8 | 95.6 | 24.4 | 38.9 | 99.8 | 52.5 | 68.8 | 81.5 | 41.4 | 53.0 | 93.5 | 53.4 | 68.0 | 94.1 | 38.8 | 54.9 | 93.4 | 40.3 | 55.1±1.4 |
| **ROS** | 88.4 | 76.7 | 82.1 | 87.5 | 77.8 | **82.4** | 99.3 | 93.0 | 96.0 | 100.0 | 99.4 | **99.7** | 74.8 | 81.2 | **77.9** | 69.7 | 86.6 | **77.2** | 86.6 | 85.8 | **85.9±0.2** |
| **VGGNet** | | | | | | | | | | | | | | | | | | | | | | |
| | A → W | | | A → D | | | D → W | | | W → D | | | D → A | | | W → A | | | **Avg.** | | |
| | OS* | UNK | **HOS** | OS* | UNK | **HOS** | OS* | UNK | **HOS** | OS* | UNK | **HOS** | OS* | UNK | **HOS** | OS* | UNK | **HOS** | OS* | UNK | **HOS** |
| OSBP [37] | 79.4 | 75.8 | 77.5 | 87.9 | 75.2 | 81.0 | 96.8 | 93.4 | **95.0** | 98.9 | 84.2 | 91.0 | 74.4 | 82.4 | **78.2** | 69.7 | 76.4 | 72.9 | 84.5 | 81.2 | 82.6±0.8 |
| **ROS** | 80.3 | 81.7 | **81.0** | 81.8 | 76.5 | 79.0 | 99.5 | 89.9 | 94.4 | 99.3 | 100.0 | **99.7** | 76.7 | 79.6 | 78.1 | 62.2 | 91.6 | **74.1** | 83.3 | 86.5 | **84.4±0.2** |
| AoD [13] | 87.7 | 73.4 | 79.9 | 92.0 | 71.1 | **79.3** | 99.8 | 78.9 | 88.1 | 99.3 | 87.2 | 92.9 | 88.4 | 13.6 | 23.6 | 82.6 | 57.3 | 67.7 | 91.6 | 63.6 | 71.9 |

**Table 2.** Accuracy (%) averaged over three runs of each method on Office-Home dataset using ResNet-50 as backbone

| **Office-Home** | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Pr → Rw | | | Pr → Cl | | | Pr → Ar | | | Ar → Pr | | | Ar → Rw | | | Ar → Cl | | | | | |
| | OS* | UNK | **HOS** | OS* | UNK | **HOS** | OS* | UNK | **HOS** | OS* | UNK | **HOS** | OS* | UNK | **HOS** | OS* | UNK | **HOS** | | | |
| $STA_{sum}$ [26] | 78.1 | 63.3 | 69.7 | 44.7 | 71.5 | 55.0 | 55.4 | 73.7 | 63.1 | 68.7 | 59.7 | 63.7 | 81.1 | 50.5 | 62.1 | 50.8 | 63.4 | 56.3 | | | |
| $STA_{max}$ [26] | 76.2 | 64.3 | 69.5 | 44.2 | 67.1 | 53.2 | 54.2 | 72.4 | 61.9 | 68.0 | 48.4 | 54.0 | 78.6 | 60.4 | 68.3 | 46.0 | 72.3 | 55.8 | | | |
| OSBP [37] | 76.2 | 71.7 | 73.9 | 44.5 | 66.3 | 53.2 | 59.1 | 68.1 | **63.2** | 71.8 | 59.8 | 65.2 | 79.3 | 67.5 | 72.9 | 50.2 | 61.1 | 55.1 | | | |
| UAN [49] | 84.0 | 0.1 | 0.2 | 59.1 | 0.0 | 0.0 | 73.7 | 0.0 | 0.0 | 81.1 | 0.0 | 0.0 | 88.2 | 0.1 | 0.2 | 62.4 | 0.0 | 0.0 | | | |
| **ROS** | 70.8 | 78.4 | **74.4** | 46.5 | 71.2 | **56.3** | 57.3 | 64.3 | 60.6 | 68.4 | 70.3 | **69.3** | 75.8 | 77.2 | **76.5** | 50.6 | 74.1 | **60.1** | | | |
| | Rw → Ar | | | Rw → Pr | | | Rw → Cl | | | Cl → Rw | | | Cl → Ar | | | Cl → Pr | | | **Avg.** | | |
| | OS* | UNK | **HOS** | OS* | UNK | **HOS** | OS* | UNK | **HOS** | OS* | UNK | **HOS** | OS* | UNK | **HOS** | OS* | UNK | **HOS** | OS* | UNK | **HOS** |
| $STA_{sum}$ | 67.9 | 62.3 | 65.0 | 77.9 | 58.0 | 66.4 | 51.4 | 57.9 | 54.2 | 69.8 | 63.2 | 66.3 | 53.0 | 63.9 | 57.9 | 61.4 | 63.5 | 62.5 | 63.4 | 62.6 | 61.9±2.1 |
| $STA_{max}$ | 67.5 | 66.7 | 67.1 | 77.1 | 55.4 | 64.5 | 49.9 | 61.1 | 54.5 | 67.0 | 66.7 | 66.8 | 51.4 | 65.0 | 57.4 | 61.8 | 59.1 | 60.4 | 61.8 | 63.3 | 61.1±0.3 |
| OSBP | 66.1 | 67.3 | 66.7 | 76.3 | 68.6 | 72.3 | 48.0 | 63.0 | 54.5 | 72.0 | 69.2 | **70.6** | 59.4 | 70.3 | **64.3** | 67.0 | 62.7 | 64.7 | 64.1 | 66.3 | 64.7±0.2 |
| UAN | 77.5 | 0.1 | 0.2 | 85.0 | 0.1 | 0.1 | 66.2 | 0.0 | 0.0 | 80.6 | 0.1 | 0.2 | 70.5 | 0.0 | 0.0 | 74.0 | 0.1 | 0.2 | 75.2 | 0.0 | 0.1±0.0 |
| **ROS** | 67.0 | 70.8 | **68.8** | 72.0 | 80.0 | **75.7** | 51.5 | 73.0 | **60.4** | 65.3 | 72.2 | 68.6 | 53.6 | 65.5 | 58.9 | 59.8 | 71.6 | **65.2** | 61.6 | 72.4 | **66.2± 0.3** |

our average OS* is similar to that of the competing methods, our average UNK is significantly higher. This characteristic is also visible qualitatively by looking at the t-SNE visualizations in Figure 4 where we focus on the comparison against the second best method OSBP. Here the features for the known (red) and unknown (blue) target data appear more confused than for ROS.

*Is it possible to reproduce the reported results of the state-of-the-art?* By analyzing the published OSDA papers, we noticed some incoherence in the reported results. For example, some of the results from OSBP are different between the pre-print [38] and the published [37] version, although they present the same description for method and hyper-parameters. Also, AoD [13] compares against the pre-print results of OSBP, while omitting the results of STA. To dissipate these ambiguities and gain a better perspective on the current state-of-the-art methods, in Table 3 we compare the results on Office-31 reported in previous works with the results obtained by running their code. For this analysis we focus on OS since it is the only metric reported for some of the methods. The comparison shows that, despite using the original implementation and the information provided by the authors, the OS obtained by re-running the experiments is between 1.3% and 4.9% lower than the originally published results. The significance of this gap calls for greater attention in providing all the relevant information for

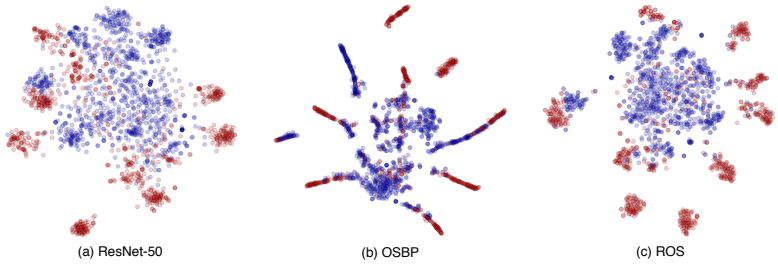



**Fig. 4.** t-SNE visualization of the target features for the W→A domain shift from Office-31. Red and blue points are respectively features of known and unknown classes

**Table 3.** Reported vs reproduced OS accuracy (%) averaged over three runs

| **Reproducibility Study** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Office-31 (ResNet-50) | | | | | | Office-31 (VGGNet) | | |
| $STA_{sum}$ | | | UAN | | | OSBP | | |
| $OS_{reported}$ | $OS_{ours}$ | 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** |

reproducing the experimental results. A more extensive reproducibility study is provided in Appendix B.

*Why is it important to use the HOS metric?* The most glaring example of why OS is not an appropriate metric for OSDA is provided by the results of UAN. In fact, when computing OS from the average (OS*,UNK) in Table 1 and 2, we can see that UAN has OS=72.5% for Office-Home and OS=91.4% for Office-31. This is mostly reflective of the ability of UAN in recognizing the known classes (OS*), but it completely disregards its (in)ability to identify the unknown samples (UNK). For example, for most domain shifts in Office-Home, UAN does not assign (almost) any samples to the unknown class, resulting in UNK=0.0%. On the other hand, HOS better reflects the open set scenario and assumes a high value only when OS* and UNK are both high.

*Is rotation recognition effective for known/unknown separation in OSDA?* To better understand the effectiveness of rotation recognition for known/unknown separation, we measure the performance of our Stage I and compare it to the Stage I of STA. Indeed, also STA has a similar two-stage structure, but uses a multi-binary classifier instead of a multi-rotation classifier to separate known and unknown target samples. To assess the performance, we compute the *area under receiver operating characteristic curve* (AUC-ROC) over the normality scores $\mathcal{N}$ on Office-31. Table 4 shows that the AUC-ROC of ROS (91.5) is significantly higher than that of the multi-binary used by STA (79.9). Table 4 also shows the performance of Stage I when alternatively removing the center loss (No Center Loss) from Equation (2) ($\lambda_{1,2} = 0$) and the anchor image (No Anchor) when training $R_1$, thus passing from relative rotation to the more standard absolute

**Table 4.** Ablation Analysis on Stage I and Stage II

| **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** |
| Multi-Binary (from STA [26]) | 83.2 | 84.1 | 86.8 | 72.0 | 75.7 | 78.3 | 79.9 |
| ROS - No Center loss | 88.8 | 83.2 | 98.8 | 99.8 | 84.7 | 84.5 | 89.9 |
| ROS - No Anchor | 84.5 | 84.9 | 99.1 | 99.9 | 87.6 | 86.2 | 90.4 |
| ROS - No Rot. Score | 86.3 | 82.7 | 99.5 | 99.9 | 86.3 | 82.9 | 89.6 |
| ROS - No Ent. Score | 80.7 | 78.7 | 99.7 | 99.9 | 86.6 | 84.4 | 88.3 |
| ROS - No Center loss, No Anchor | 76.5 | 79.1 | 98.3 | 99.7 | 85.2 | 83.5 | 87.1 |
| ROS - No Rot. Score, No Anchor | 83.9 | 84.6 | 99.4 | 99.9 | 84.7 | 84.9 | 89.6 |
| ROS - No Ent. Score, No Anchor | 80.1 | 81.0 | 99.5 | 99.7 | 84.3 | 83.3 | 87.9 |
| ROS - No Rot. Score, No Center loss | 80.9 | 81.6 | 98.9 | 99.8 | 85.6 | 83.3 | 88.3 |
| ROS - No Ent. Score, No Center loss | 76.4 | 79.8 | 99.0 | 98.3 | 84.4 | 84.3 | 87.0 |
| ROS - No Ent. Score, No Center loss, No Anchor | 78.6 | 80.4 | 99.0 | 98.9 | 86.2 | 83.2 | 87.7 |
| ROS - No Rot. Score, No Center loss, No Anchor | 78.7 | 82.2 | 98.3 | 99.8 | 85.0 | 82.6 | 87.8 |
| **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 Stage I - GRL [14] Stage II | 83.5 | 80.9 | 97.1 | 99.4 | 77.3 | 72.6 | 85.1 |
| ROS Stage I - No Anchor in Stage II | 80.0 | 82.3 | 94.5 | 99.2 | 76.9 | 76.6 | 84.9 |
| ROS Stage I - No Anchor, No Entropy in Stage II | 80.1 | 84.4 | 97.0 | 99.2 | 76.5 | 72.9 | 85.0 |

rotation. In both cases, the performance significantly drops compared to our full method, but still outperforms the multi-binary classifier of STA.

*Why is the normality score defined the way it is?* As defined in Equation (3), our normality score is a function of the rotation score and entropy score. The rotation score is based on the ability of $R_1$ to predict the rotation of the target samples, while the entropy score is based on the confidence of such predictions. Table 4 shows the results of Stage I when alternatively discarding either the rotation score (No Rot. Score) or the information of the entropy score (No Ent. Score). In both cases the AUC-ROC significantly decreases compared to the full version, justifying our choice.

*Is rotation recognition effective for domain alignment in OSDA?* While rotation classification has already been used for CSDA [47], its application in OSDA, where the shared target distribution could be noisy (*i.e.* contain unknown samples) has not been studied. On the other hand, GRL [14] is used, under different forms, by all existing OSDA methods. We compare rotation recognition and GRL in this context by evaluating the performance of our Stage II when replacing the $R_2$ with a domain discriminator. Table 4 shows that rotation recognition performs on par with GRL, if not slightly better. Moreover we also evaluate the role of the relative rotation in the Stage II: the results in the last row of Table 4 confirm that it improves over the standard absolute rotation (No Anchor in Stage II) even when the rotation classifier is used as cross-domain adaptation strategy. Finally, the cosine distance between the source and the target domain without adaptation in Stage II (0.188) and with our full method (0.109) confirms that rotation recognition is indeed helpful to reduce the domain gap.

*Is our method effective on problems with a high degree of openness?* The standard open set setting adopted in so far, presents a relatively balanced number of

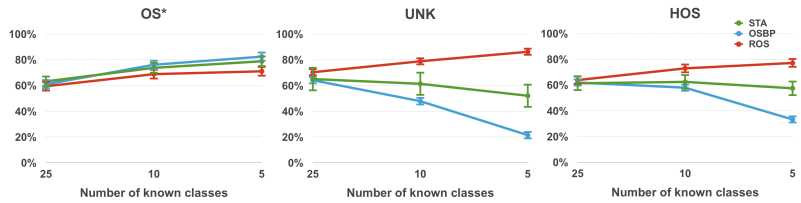



**Fig. 5.** Accuracy (%) averaged over the three openness configurations.

shared and private target classes with openness close to 0.5. Specifically it is $\mathbb{O} = 1 - \frac{10}{21} = 0.52$ for Office-31 and $\mathbb{O} = 1 - \frac{25}{65} = 0.62$ for Office-Home. In real-world problems, we can expect the number of unknown target classes to largely exceed the number of known classes, with openness approaching 1. We investigate this setting using Office-Home and, starting from the classes sorted with ID from 0 to 64 in alphabetic order, we define the following settings with increasing openness: **25** known classes $\mathbb{O} = 0.62$, ID:{0-24, 25-49, 40-64}, **10** known classes $\mathbb{O} = 0.85$, ID:{0-9, 10-19, 20-29}, **5** known classes $\mathbb{O} = 0.92$, ID:{0-4, 5-9, 10-14}. Figure 5 shows that the performance of our best competitors, STA and OSBP, deteriorates with larger $\mathbb{O}$ due to their inability to recognize the unknown samples. On the other hand, ROS maintains a consistent performance.

## 6 Discussion and conclusions

In this paper, we present ROS: a novel method that tackles OSDA by using the self-supervised task of predicting image rotation. We show that, with simple variations of the rotation prediction task, we can first separate the target samples into known and unknown, and then align the target samples predicted as known with the source samples. Additionally, we propose HOS: a new OSDA metric defined as the harmonic mean between the accuracy of recognizing the known classes and rejecting the unknown samples. HOS overcomes the drawbacks of the current metric OS where the contribution of the unknown classes vanishes with increasing number of known classes.

We evaluate the perfomance of ROS on the standard Office-31 and Office-Home benchmarks, showing that it outperforms the competing methods. In addition, when tested on settings with increasing openness, ROS is the only method that maintains a steady performance. HOS reveals to be crucial in this evaluation to correctly assess the performance of the methods on both known and unknown samples. Finally, the failure in reproducing the reported results of existing methods exposes an important issue in OSDA that echoes the current reproducibility crisis in machine learning. We hope that our contributions can help laying a more solid foundation for the field.

**Acknowledgements** This work was partially founded by the ERC grant 637076 RoboExNovo (SB), by the H2020 ACROSSING project grant 676157 (MRL) and took advantage of the NVIDIA GPU Academic Hardware Grant (TT).

# Appendix

## A 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 [36], Office-Home [44] and using two different backbones, ResNet-50 [18] and VGGNet [41]. 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 [9] 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 [22] 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).

*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 [9] 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.

**Table 5.** Office-Home Resnet50

| Office-Home | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Pr → Rw | | | | Pr → Cl | | | |
| | OS | OS* | UNK | **HOS** | OS | OS* | UNK | **HOS** |
| $STA_{sum}$ [26] | 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 |
| $STA_{max}$ [26] | 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 [37] | 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 [49] | 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** |

*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 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

**Table 6.** Office-31

| Office-31 ResNet-50 | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | A → W | | | | A → D | | | | | | | |
| | OS | OS* | UNK | **HOS** | OS | OS* | UNK | **HOS** | | | | |
| $STA_{sum}$ [26] | 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 | | | | |
| $STA_{max}$ [26] | 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 [37] | 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 [49] | 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 [37] | 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 [13] | 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 [13] | 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 [13] | 81.6 | 88.4 | 13.6 | 23.6 | 80.3 | 82.6 | 57.3 | 67.7 | 89.1 | 91.6 | 63.6 | 71.9 |

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.

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 5 and 6. We remark that, in terms of OS and OS*, STA is extremely unstable with large standard deviations over multiple runs.

## B 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 7 we compare the results published in the official papers of STA [26], OSBP [37], and UAN [49] considering the OS accuracy since it is the only metric shared by all the works. For UAN we replicated the particular settings described in

**Table 7.** 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** |

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* [26] `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* [37] `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 [33] 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).

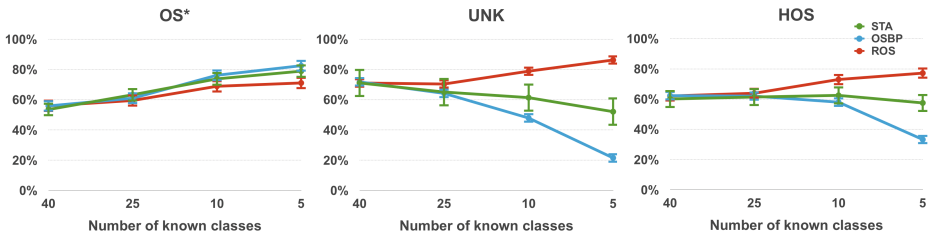
*UAN* [49] `https://github.com/thuml/Universal-Domain-Adaptation`
This repository provides the code for UAN with ResNet-50 as backbone: specific files contain instructions to run experiments on both Office-31 and Office-Home. On Office-31 we trained for 20000 iterations with batch size 36, SGD with and 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.

## C 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 $\mathbb{O} = 1 - \frac{40}{65} = 0.38$ using ID:{0-39, 15-54, 25-64}. The results in Figure 6 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.

## D 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 7. 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 [5,26,48]. 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




**Office-Home (Avg.)**

| | OS | OS* | UNK | **HOS** | OS | OS* | UNK | **HOS** |
|---|---|---|---|---|---|---|---|---|
| | 40 known classes | | | | 25 known classes | | | |
| $STA_{sum}$[26] | 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[37] | 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** |
| $STA_{sum}$[26] | 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[37] | 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. 6.** 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

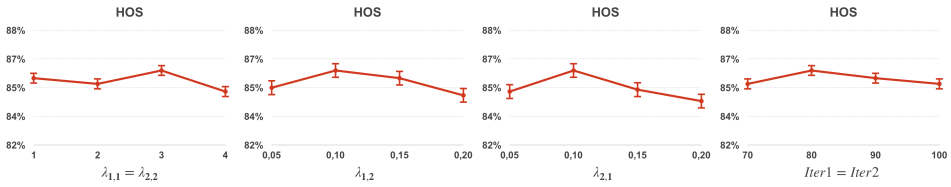



**Fig. 7.** Hyper-parameter analysis

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 [13]).

## E 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 [47,5] and anomaly detection [17,2,21], rotation classification clearly emerges as the most reliable candidate for our purpose. To confirm our claim, we run additional experiments on Office-31

**Table 8.** 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 9.** 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[26] | UAN[49] | OSBP[37] | **ROS** |
| 1069s | 9615s | 3672s | 1875s |

(ResNet-50) with alternative self-supervised tasks. Following [17], we considered the self-supervised task of translation classification for anomaly detection (Stage I). Moreover, following [5], we considered the self-supervised task of solving a jigsaw puzzle for domain alignment (Stage II). Table 8 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 8 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 8: 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. Indeed, the results in in Table 8 indicate that adding the center loss to Stage II might even produce a slight drop in performance.

## F 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 9 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).

## G 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.

**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**
      $\mathcal{D}_t^{unk} \leftarrow \boldsymbol{x}_j$

## References


1. Bendale, A., Boult, T.E.: Towards open set deep networks. In: CVPR (2016)
2. Bergman, L., Hoshen, Y.: Classification-based anomaly detection for general data. In: ICLR (2020)
3. Bousmalis, K., Trigeorgis, G., Silberman, N., Krishnan, D., Erhan, D.: Domain Separation Networks. In: NeurIPS (2016)
4. Candès, E.J., Li, X., Ma, Y., Wright, J.: Robust principal component analysis? J. ACM **58**(3) (Jun 2011)
5. Carlucci, F.M., D'Innocente, A., Bucci, S., Caputo, B., Tommasi, T.: Domain generalization by solving jigsaw puzzles. In: CVPR (2019)
6. Reproducibility Challenge. `https://reproducibility-challenge.github.io/neurips2019/`, accessed: 4-3-2020
7. The Machine Learning Reproducibility Checklist. `https://www.cs.mcgill.ca/~jpineau/ReproducibilityChecklist.pdf`, accessed: 4-3-2020
8. Csurka, G.: Domain Adaptation in Computer Vision Applications. Springer Publishing Company, Incorporated, 1st edn. (2017)
9. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: CVPR (2009)
10. Dodge, J., Gururangan, S., Card, D., Schwartz, R., Smith, N.A.: Show your work: Improved reporting of experimental results. In: EMNLP (2019)
11. Doersch, C., Gupta, A., Efros, A.A.: Unsupervised visual representation learning by context prediction. In: ICCV (2015)
12. Eskin, E., Arnold, A., Prerau, M., Portnoy, L., Stolfo, S.: A geometric framework for unsupervised anomaly detection. In: Applications of data mining in computer security, pp. 77–101. Springer (2002)
13. Feng, Q., Kang, G., Fan, H., Yang, Y.: Attract or distract: Exploit the margin of open set. In: ICCV (2019)
14. Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., Lempitsky, V.: Domain-adversarial training of neural networks. The Journal of Machine Learning Research **17**(1), 2096–2030 (2016)
15. Ghifary, M., Kleijn, W.B., Zhang, M., Balduzzi, D., Li, W.: Deep reconstruction-classification networks for unsupervised domain adaptation. In: ECCV (2016)
16. Gidaris, S., Singh, P., Komodakis, N.: Unsupervised representation learning by predicting image rotations. arXiv preprint arXiv:1803.07728 (2018)
17. Golan, I., El-Yaniv, R.: Deep anomaly detection using geometric transformations. In: NeurIPS (2018)
18. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016)
19. Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., Meger, D.: Deep reinforcement learning that matters. In: AAAI (2018)
20. Hendrycks, D., Gimpel, K.: A baseline for detecting misclassified and out-of-distribution examples in neural networks. In: ICLR (2017)
21. Hendrycks, D., Mazeika, M., Kadavath, S., Song, D.: Using self-supervised learning can improve model robustness and uncertainty. NeurIPS (2019)
22. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. Preprint arXiv:1502.03167 (2015)
23. Kim, J., Scott, C.D.: Robust kernel density estimation. J. Mach. Learn. Res. **13**(1), 25292565 (Sep 2012)

24. Larsson, G., Maire, M., Shakhnarovich, G.: Colorization as a proxy task for visual understanding. In: CVPR (2017)
25. Liang, S., Li, Y., Srikant, R.: Enhancing the reliability of out-of-distribution image detection in neural networks. In: ICLR (2018)
26. Liu, H., Cao, Z., Long, M., Wang, J., Yang, Q.: Separate to adapt: Open set domain adaptation via progressive separation. In: CVPR (2019)
27. Loghmani, M.R., Vincze, M., Tommasi, T.: Positive-unlabeled learning for open set domain adaptation. Pattern Recognition Letters **136**, 198 – 204 (2020)
28. Long, M., Cao, Y., Wang, J., Jordan, M.I.: Learning transferable features with deep adaptation networks. In: ICML (2015)
29. Lucic, M., Kurach, K., Michalski, M., Gelly, S., Bousquet, O.: Are gans created equal? a large-scale study. In: NeurIPS (2018)
30. Noroozi, M., Favaro, P.: Unsupervised learning of visual representations by solving jigsaw puzzles. In: ECCV (2016)
31. Panareda Busto, P., Gall, J.: Open set domain adaptation. In: ICCV (2017)
32. Pathak, D., Krähenbühl, P., Donahue, J., Darrell, T., Efros, A.: Context encoders: Feature learning by inpainting. In: CVPR (2016)
33. Peng, X., Usman, B., Kaushik, N., Hoffman, J., Wang, D., Saenko, K.: Visda: The visual domain adaptation challenge. Preprint arXiv:1710.06924 (2017)
34. Ruff, L., Vandermeulen, R., Goernitz, N., Deecke, L., Siddiqui, S.A., Binder, A., Müller, E., Kloft, M.: Deep one-class classification. In: ICML (2018)
35. Russo, P., Carlucci, F.M., Tommasi, T., Caputo, B.: From source to target and back: symmetric bi-directional adaptive gan. In: CVPR (2018)
36. Saenko, K., Kulis, B., Fritz, M., Darrell, T.: Adapting visual category models to new domains. In: ECCV (2010)
37. Saito, K., Yamamoto, S., Ushiku, Y., Harada, T.: Open set domain adaptation by backpropagation. In: ECCV (2018)
38. Saito, K., Yamamoto, S., Ushiku, Y., Harada, T.: Open set domain adaptation by backpropagation. arXiv preprint arXiv:1804.10427 (2018)
39. Schlegl, T., Seeböck, P., Waldstein, S.M., Schmidt-Erfurth, U., Langs, G.: Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. In: IPMI (2017)
40. Schölkopf, B., Williamson, R., Smola, A., Shawe-Taylor, J., Platt, J.: Support vector method for novelty detection. In: NeurIPS (1999)
41. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
42. Sun, B., Feng, J., Saenko, K.: Return of frustratingly easy domain adaptation. In: AAAI (2016)
43. Tzeng, E., Hoffman, J., Saenko, K., Darrell, T.: Adversarial discriminative domain adaptation. In: CVPR (2017)
44. Venkateswara, H., Eusebio, J., Chakraborty, S., Panchanathan, S.: Deep hashing network for unsupervised domain adaptation. In: CVPR (2017)
45. Wen, Y., Zhang, K., Li, Z., Qiao, Y.: A discriminative feature learning approach for deep face recognition. In: ECCV (2016)
46. Xia, Y., Cao, X., Wen, F., Hua, G., Sun, J.: Learning discriminative reconstructions for unsupervised outlier removal. In: ICCV (2015)
47. Xu, J., Xiao, L., Lpez, A.M.: Self-supervised domain adaptation for computer vision tasks. IEEE Access **7**, 156694–156706 (2019)
48. Xu, R., Li, G., Yang, J., Lin, L.: Larger norm more transferable: An adaptive feature norm approach for unsupervised domain adaptation. In: ICCV (2019)

49. You, K., Long, M., Cao, Z., Wang, J., Jordan, M.I.: Universal domain adaptation. In: CVPR (2019)
50. Zhai, S., Cheng, Y., Lu, W., Zhang, Z.: Deep structured energy based models for anomaly detection. In: ICML (2016)
51. Zhang, R., Isola, P., Efros, A.A.: Colorful image colorization. In: ECCV (2016)
52. Zhou, C., Paffenroth, R.C.: Anomaly detection with robust deep autoencoders. In: ACM SIGKDD (2017)
53. Zimek, A., Schubert, E., Kriegel, H.P.: A survey on unsupervised outlier detection in high-dimensional numerical data. Statistical Analysis and Data Mining: The ASA Data Science Journal **5**, 363–387 (2012)
54. Zong, B., Song, Q., Min, M.R., Cheng, W., Lumezanu, C., Cho, D., Chen, H.: Deep autoencoding gaussian mixture model for unsupervised anomaly detection. In: ICLR (2018)