Open-Set Domain Adaptation through Self-Supervision

Protopapa Andrea, Quarta Matteo, Ruggeri Giuseppe, Versace Alessandro
Politecnico di Torino
Italy
s{286302,292477,292459,292435}@studenti.polito.it

Abstract

In machine learning applications, domain adaptation (DA) techniques try to mitigate the problem of having different domains in the training and test data. Another common problem is represented by the presence of more semantic classes in the test data, which are unknown and completely new to the developed models. The latter problem comes under the name of novelty or anomaly detection. In real world scenarios, it is becoming extremely common suffering of both problems. Open-Set Domain Adaptation (OSDA) methods try to tackle these problems by jointly adapting a model trained on a labeled source domain to an unlabeled target domain while performing novelty detection. We propose a new method leveraging a self-supervised technique, rotation recognition, constisting in first performing novelty detection on the target data and then aligning the two domains avoiding potential negative adaptation. Furthermore, we assess the performance using a new metric which represents in a balanced way the ability to jointly solve the two problems. Experiments conducted on the Office-Home benchmark show interesting results and method effectiveness. Full code for our proposed method is made publicly available https://github.com/coccocarmiano/AML2021

1. Introduction

Nowadays, the widespread usage of deep neural networks to accomplish computer vision tasks has brought huge benefits. In real world applications, as the tasks to cope with are becoming more and more challenging, machine learning methods commonly suffer of a gap between the performance obtained during the development, and the actual performance observed in real usages.

One of the problems which is causing this loss of performance is the domain gap between the training data and the actual observed data. Intuitively, if we train a model on a specific domain, such as employing real world pictures depicting real objects, for example to perform classification, we expect that the model will perform well on a fairly large variety of test cases. However, the actual data present a huge variety on the domains while still representing the same semantic classes that we want to predict. For example, we may want to be capable of predicting that both an image of a real elephant and a drawing of an elephant are containing the semantic class *elephant*. Domain adaptation techniques have been developing in recent years to reduce the domain gap between a labeled source domain and one or more unlabeled target domains. Generally, this is done by enforcing the learning of domain-invariant patterns of both domains.

Another big issue, named *O*pen-Set Recognition (OSR) [16], features the presence of additional anomalous semantic classes in the observed data and requires to both accurately classify known observed samples and also reject the ones from unknown classes.

The joint presence and accounting of these two problems has been formalized as a new sub-field of computer vision with the name of *Open-Set Domain Adaptation (OSDA)*. As a consequence, if we try to reduce the domain gap between the whole target and the source domain, we will observe an unwanted alignment between the data belonging to anomalous semantic classes and the source classes we want to model and predict. For this reason, it is important to first perform anomaly detection of the additional set of unknown classes and then do the alignment between the source and the target domain identified as known translating the problem into a *Closed-Set Domain Adaptation (CSDA)* one.

Common machine learning methods usually leverage huge manually annotated datasets to perform well on the given tasks. However, acquiring such data is often very costly and relying on this data may not be scalable in large applications on the long run. Recently, a commonly employed approach is self-supervised learning, which consists in creating new automatically labeled data starting from the original unlabeled data. The fundamental idea is creating some auxiliary task from input data so that the model can learn the underlying structure of the data, such as high-level knowledge, correlations, and metadata embedded. This type of learning has been recently used for Domain Adapta-

tion, learning robust cross-domain features and supporting generalization [7, 38], and also for some Open Set problems specialized in anomaly detection and discriminating anomalous data [2, 18].

The approach presented in this paper combines the power of the self-supervised learning with the standard supervised learning approach for semantic class recognition. We propose a two stage method aiming to identify and isolate unknown class samples in the first stage, and reducing the domain gap between domains in the second stage to avoid negative transfer. A schematic view of the stages can be seen in figures 1, 2, 3. This is done by using a modified version of the rotation task as self-supervised method, predicting the relative rotation between an image and its rotated version. Finally, a classifier is used to predict if each target sample belongs to either one of the knwon classes or to an unknown class, being rejected in the latter case. We evaluate the method on the Office-Home benchmark [36] using a new OSDA metric.

To wrap up, our **main contributions** are:

1. we define a new method to tackle OSDA problems which exploits the rotation recognition task to perform both the known/unknown target separation and the domain adaptation;

2. we introduce a new OSDA metric which properly balances the measure of both the performance on predicting the known classes and the performance on doing the unknown rejection;

3. we conduct an extensive ablation over the hyperparameters for different variants of the self-supervised task underlying the benefits of some techniques over others.

2. Related Work

Closed-Set Domain Adaptation is the setting where labeled training data is available on a source domain, but the goal is to have good performance on a unlabeled target domain with a different marginal distribution of data. The aim is to align the learned representations of the source and target domains.

Methods for unsupervised domain adaptation in computer vision can be divided into three broad classes. The first class aims to induce alignment in some feature space optimizing the distributional discrepancy [4, 14, 21, 26]. The second class of methods directly transforms the source images to resemble the target images with generative models [3, 34, 35], operating on image pixels directly instead of an intermediate representation space. The third class uses a model trained on the labeled source data to estimate labels on the target data, then trains on some of those estimated pseudo-labels (e.g. the most confident ones), therefore bootstrapping through the unlabeled target data. This technique is borrowed from semi-supervised learning, where it is called co-training [8, 30, 42].

In contrast, the presented method uses a Self-Supervised approach, already presented in the recent years by many other works [29, 33, 40] using pretext tasks (e.g., image rotation, jigsaw puzzle [7], mutual information (MI) [13], instance discrimination [9]) to learn high-level feature representation in source and target domains by jointly training a shared feature extractor.

Open Set Recognition (OSR) has the objective of learning a classifier that can reject the unknown samples while classifying the known classes accurately.

The problem was first formulated in [32] and since then, several other works have analyzed this challenge in the context of deep networks [1, 20, 28]. Some techinques presented in the past were focus on an adversarial approach to delineate closed from open-set images [15, 24].

In the recent years different works have landed to a Self-Supervised solution, used also in the presented method, engaging in learning invariant representations to the transformations of the input data, improving separation of classes from each other and from open-set samples [10, 18, 22].

Self-supervised Learning is an emerging field focused on using data itself as supervision for auxiliary (also called "pretext") tasks that learn deep feature representations, which will hopefully be informative for downstream "real" tasks. Many such auxiliary tasks have been proposed in the literature, including image colorization [41], image jigsaw puzzle [27], geometric transformations [11].

Another geometric transformation task is rotation recognition [17], where input images are rotated by multiples of 90° and the network is trained to predict the rotation angle of each image. This pretext task has been successfully used for both anomaly detection [18] and closed-set domain adaptation [38].

Open Set Domain Adaptation (OSDA) is a challenging domain adaptation setting which allows the existence of unknown classes on the target domain. The term "OSDA" was first introduced by Busto and Gall [6], but the currently accepted definition was introduced by Saito *et al*. [31] considering the target as containing all the source categories and additional set of private categories that should be considered unknown. They have presented Open Set Back-Propagation (OSBP) [31] as an adversarial method to increase the prediction variances so that the generator can choose if accept or not a target saple. Separate To Adapt (STA) [25] train a multi-binary classifier to progressively separate the samples of unknown and known classes. Universal Adaptation Network (UAN) [39], originally proposed for the universal domain adaptation setting that is a superset of OSDA, quantify the sample-level transferability and recognize the unknown samples based on it. Attract or Distract (AoD) [12] refine

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. Jing *et al*. [23] have recently proposed a method to recognize the unknown samples according to the centroid deviation angles and employing the statistical Extreme Value Theory to recognize the unknown samples that are misclassified into known classes.

The approach that we are going to present is instead based on a Self-Supervision task, the rotation recognition, used to both separate known and unknown target samples and align the known source and target distributions.

3. Method

3.1. Problem Formulation

We define as $\mathcal{D}_s = \{(\mathbf{x}_i^s, y_i^s)\}_{i=1}^{N_s} \sim p_s$ the source dataset whose distribution of samples and labels is p_s, while $\mathcal{D}_t = \{\mathbf{x}_i^t\}_{i=1}^{N_t} \sim p_t$ is the unlabeled target dataset drawn from distribution p_t.

The source dataset $\mathcal{D}_s$ is associated with a set of known classes $\mathcal{C}_s$, whereas the target dataset $\mathcal{D}_t$ contains a set of classes $\mathcal{C}_t = \mathcal{C}_s \cup \mathcal{C}_{t \setminus s}$. In other words, $|\mathcal{C}_s| < |\mathcal{C}_t|$ and $\mathcal{C}_s \subset \mathcal{C}_t$.

In OSDA we have that $p_s \neq p_t$. Moreover, it holds that $p_s \neq p_t^{\mathcal{C}_s}$, where $p_t^{\mathcal{C}_s}$ denotes the distribution of the target domain if we restrict to the shared classes $\mathcal{C}_s$.

Summarizing, in OSDA tasks, we have both a domain gap ($p_s \neq p_t^{\mathcal{C}_s}$) and a category gap ($\mathcal{C}_s \neq \mathcal{C}_t$). Moreover, the goal is to assign the target samples either to a category $i \in \mathcal{C}_s$, or to reject them as *unknown*. A metric to measure the complexity of an OSDA problem is the *openness* betweeen the source and the target domain [1], defined as $\mathbb{O} = 1 - \frac{\mathcal{C}_s}{\mathcal{C}_t}$. When $\mathbb{O} > 0$, we are dealing with an OSDA problem, otherwise we are in a CSDA setting.

3.2. Approach

The proposed method is split in two sequential stages. First, to avoid negative transfer during the domain alignment step, we want a model that is able to separate the target dataset into $\mathcal{D}_t^{unk}$, which contains images belonging to unknown classes, and $\mathcal{D}_t^{knw}$, which contains only images belonging to the known classes. To do that, we leverage the power of the rotation pre-text task to perform the separation. Next, in the second stage, we can close the gap between the source domain and the target domain exploiting $\mathcal{D}_t^{knw}$ using the same self-supervised task. Furthermore, we leverage $\mathcal{D}_t^{unk}$ to learn the additional *unknown* class.

3.3. Rotation Recognition

We denote with $rot(\mathbf{x}, k)$ the rotating function of a sample image $\mathbf{x}$ by $k \times 90$ degrees clockwise. The self-supervised pre-text task consists in generating a random rotation index $k \in [0, 3]$ that then becomes the label for the rotated version of the image $\tilde{\mathbf{x}} = rot(\mathbf{x}, k)$. Then, the task becomes a standard classification of the correct rotation index ($\mathcal{C}_r = \{0, 1, 2, 3\}$).

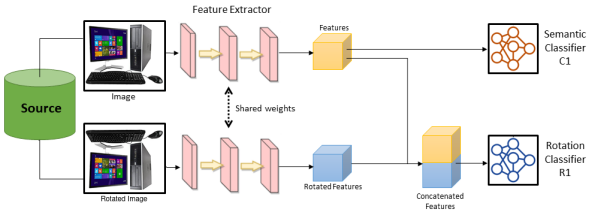

Figure 1. Schema of the learning process for the model in Step I.

Relative orientation: more precisely, we exploit a relative rotation task, which implies that both the features of the original and rotated image are supplied to the rotation classifier. This is preferred over the absolute rotation task as some objects might not have an absolute coherent orientation inside the dataset (e.g. a pen may be present in different rotated versions inside the dataset).

Multi-head rotation classifier: alternatively, instead of having a single rotation head predicting the rotation of a sample regardless of its semantic class, we also try using a different head for each known class $\in \mathcal{C}_s$, each one responsible of predicting the rotation of the images belonging to that semantic class. This variation can mitigate the problem of trying to predict the rotation of a larger number of semantic classes. Infact, as the number of semantic classes grows, the problem of predicting the relative orientation becomes more difficult.

The application of the rotation recognition pre-text task allows to effectively favor and force the model to learn domain-independent patterns, which are crucial to perform the novelty detection in a cross-domain fashion and, moreover, to successfully perform the domain alignment. To provide an explanation of why this applies, we can think that a rotation classifier needs to focus on discriminative patterns to successfully perform the rotation predictions, such as shapes, edges, and high-level object relative position like the position of the eyes w.r.t nose. The method and its effectiveness is further illustrated in [17], and further possible improvements will be discussed in section 5.

3.4. Step I: target known/unknown separation

To perform the target separation we train a CNN iterating on $\tilde{\mathcal{D}}_s = \{(\mathbf{x}_i^s, \tilde{\mathbf{x}}_i^s, z_i^s)\}_{i=1}^{N_s}$, where $\tilde{\mathbf{x}}_i^s$ is the $z_i^s \times 90$ degrees rotated version of $\mathbf{x}_i^s$. The CNN is made of a feature extractor E and two heads: R_1 and C_1. C_1 is the object classifier, which assigns to image $\mathbf{x}_i^s$ a predicted semantic class label, while R_1 is the relative rotation classifier, which assigns to

the rotated image $\tilde{\mathbf{x}}_i^s$ a predicted rotation label. To keep the notation clear, we define as $\mathbf{y_i}$ and $\mathbf{z_i}$ the one-hot vector representations of the corresponding scalar labels. Notice that the multi-head rotation classifier internally uses $|\mathcal{C}_s|$ different heads for the rotation task. In this case, the head selected to perform the rotation prediction is up to the object classifier C_1 (during inference and not during training).

The object class vector of predicted probabilities is computed as $\hat{\mathbf{y}}_\mathbf{i}^\mathbf{s} = softmax(C_1(E(\mathbf{x}_\mathbf{i}^\mathbf{s})))$, while the vector of predicted probabilities for rotation label is computed from the stacked features of the original and rotated image $\hat{\mathbf{z}}_\mathbf{i}^\mathbf{s} = softmax(R_1([E(\mathbf{x}_\mathbf{i}^\mathbf{s}), E(\tilde{\mathbf{x}}_\mathbf{i}^\mathbf{s})]))$. The model is trained to minimize the objective function $\mathcal{L}_1 = \mathcal{L}_{C_1} + \mathcal{L}_{R_1}$. This is the sum of two cross-entropy loss functions:

$$\mathcal{L}_{C_1} = -\sum_{i \in \mathcal{D}_s} \mathbf{y}_i^s \log \hat{\mathbf{y}}_i^s \tag{1}$$

$$\mathcal{L}_{R_1} = -\alpha_1 \sum_{i \in \tilde{\mathcal{D}}_s} \mathbf{z}_i^s \log \hat{\mathbf{z}}_i^s \tag{2}$$

Where α_1 is a weight associated to the rotation task. We also try using an extended rotation loss function $\mathcal{L}_{R_1}^*$ implementing an additional center loss [37] term:

$$\mathcal{L}_{R_1}^* = \sum_{i \in \tilde{\mathcal{D}}_s} -\alpha_1 \mathbf{z}_i^s \log \hat{\mathbf{z}}_i^s + \lambda ||\mathbf{v}_i^s - \gamma(\mathbf{z}_i^s)||_2^2 \tag{3}$$

Here $\mathbf{v_i}$ is the output of the penultimate layer of R_1, $\gamma(\mathbf{z}_i)$ is the centroid of the features associated to class i (notice that the centroid is relative to a different rotation class i in the multi-head variant), $||\cdot||_2^2$ is the l-2 norm and λ is the weight associated with the center loss term. The additional center loss contribution should favor having hidden features which have a smaller intra-class variance, resulting in a more discriminative representation when combined with the standard cross-entropy loss, which in contrast tries to enlarge the distance between the means of the classes.

When training is completed, we can start separating the target samples into known and unknown. To do so, *normality scores* $\mathcal{N}(\cdot)$ are used, defined as the maximum prediction of the rotation classifier: $\mathcal{N}(\tilde{\mathbf{x}}_i) = \max(\hat{\mathbf{z}}_i)$. Notice that, for each target sample, all the four rotations are applied and the resulting normality score for the sample is computed as the mean of the four normality scores. To decide if a target sample belongs to a known semantic class or not, we compare the normality score with a threshold $\tilde{\mathcal{N}}$.

$$\begin{cases} \mathbf{x}_i^t \in \mathcal{D}_t^{knw} & \text{if } \mathcal{N}(\tilde{\mathbf{x}}_i^t) \geq \tilde{\mathcal{N}} \\ \mathbf{x}_i^t \in \mathcal{D}_t^{unk} & \text{if } \mathcal{N}(\tilde{\mathbf{x}}_i^t) < \tilde{\mathcal{N}} \end{cases} \tag{4}$$

When using a multi-head rotation classifier, it is required to choose among the $|\mathcal{C}_s|$ possible heads to make the prediction. Head $R_{1,j}$ is used where $j = \arg\max_j \{\hat{\mathbf{y}}_{i,[j]}^t\}_{j=0}^{|\mathcal{C}_s|-1}$ (j is the component j of the vector).

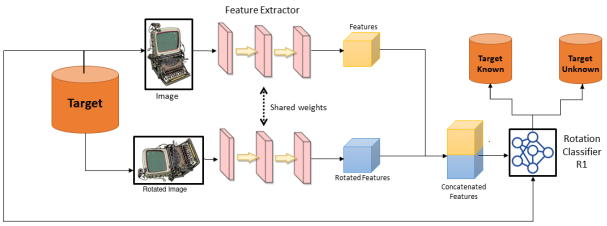

Figure 2. Schema of the separation process into known and unkown samples on target images

The key idea behind the normality score is that, if R_1 is confident enough on its predicted rotation, it is likely that it has managed to successfully recognize the rotation. Since R_1 has learned the domain-independent patterns of the known classes from the source dataset up to this point, it should be able to recognize the rotations applied only to images belonging to such classes.

3.5. Step II: domain alignment

In this step, having separated the target into a known part $\mathcal{D}_t^{knw}$, and an unknown part $\mathcal{D}_t^{unk}$, we arrange two new datasets in order to perform the domain alignment while also learning the unknown class. The first one is $\mathcal{D}_s^*$, composed as $\mathcal{D}_s \cup \mathcal{D}_t^{unk}$, which contains the original source images plus the target images identified as unknown classes. We thus set the labels for $\mathcal{D}_t^{unk}$ as the class *unknown*. The second one is $\mathcal{D}_t^{knw}$, which can be used to perform the domain alignment without the risk of negative transfer. While the feature extractor E is inherited from the previous stage and leverages the previous training phase, we also use two new classifiers, C_2 and R_2. They are similar to the previous classifiers but they have two important differences. C_2 now has a $(|\mathcal{C}_s| + 1)$-dimensional output to accomodate also the unknown class predictions and also benefits from the previous learning, while R_2 is always a single-head rotation classifier and starts the learning from scratch. The training phase is the same as before with the difference that we do not have the center loss this time. We also employ a different hyperparameter α_2 to weigh the rotation classifier loss contribution. The objective function is again $\mathcal{L}_2 = \mathcal{L}_{C_2} + \mathcal{L}_{R_2}$, where the two contributions are identical to equations 1 and 2. We report the $\mathcal{L}_{R_2}$ loss contribution to make clear the usage of α_2 and we recall that $\mathcal{L}_{C_2}$ is now computed using the new arranged dataset $\mathcal{D}_s^*$.

$$\mathcal{L}_{R_2} = -\alpha_2 \sum_{i \in \mathcal{D}_t^{knw}} \mathbf{z}_i^s \log \hat{\mathbf{z}}_i^s \tag{5}$$

3.6. Performance metrics

To have a meaningful comparison of the effectiveness of the models to solve OSDA problems, we need to resort to

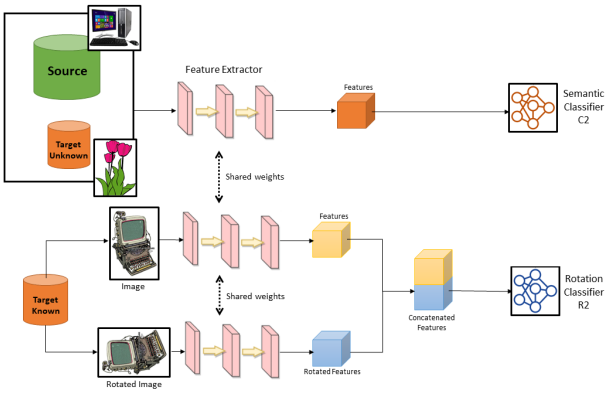

Figure 3. Schema of the learning process of the model in Step II

a metric which correctly measures the goodness of our predictions. The difficulty lies on the joint presence of two, usually conflicting, goals. We recall that an OSDA problem is characterized by the concurrent presence of a domain difference and a class set difference. The final goal is to correctly classify the samples belonging to known classes while rejecting the samples of unknown classes. The performance of both the goals are typically taken into account by two metrics, *OS**, which is the fraction of the correctly classified samples belonging to known classes, and *UNK*, which in contrast is the accuracy over the correctly rejected samples. Since the goals are usually conflicting, we need an overall metric which is able to correctly balance the two contributions, providing low values if at least one of the two is low. We therefore pick the harmonic mean of the two contributions, defined as $HOS = 2\frac{OS^* \times UNK}{OS^* + UNK}$. We underline that HOS is a more severe and fair metric to effectively measure the goodness of a model to perform well on a given OSDA problem because, from the definition of the harmonic mean, it tends to give a bigger weight to the smaller values.

Lastly, we also exploit the AUROC metric to assess the goodness of the system in separating the target into a known and unknown part.

4. Experiments

4.1. Benchmark

We evaluate the effectiveness of our proposed method exploiting the *Office-Home* [36] benchmark dataset, widely used to assess the performance for OSDA settings. The dataset is characterized by four different domains: Art (A), Clipart (C), Product (P) and Real World (R), and each domain contains the same 65 object class categories. We underline that the large domain gaps and the number of classes make the benchmark challenging. For the experiments, we fix the first 45 classes (in alphabetical order) as the known classes ($\mathcal{C}_s$), and the remaining 20 classes as the unknown ones ($\mathcal{C}_{t \setminus s}$). For each experiment, we collect the HOS value achieved after the last training epoch of the step 2, along with the AUROC value computed after the step 1, crucial to assess the goodness of the system's components responsible for the separation of the target.

4.2. Implementation Details

As we have already seen in section 3, the network is composed of a feature extractor E, and two heads, $C_{1/2}$ and $R_{1/2}$. For the feature extraction, we employ a ResNet-18 [19] pre-trained on ImageNet. All the experiments are run training with a batch size equals to 32 and using a stochastic gradient descent optimization strategy with a weight decay of 0.0005 and a momentum of 0.9. The learning rate is set to 0.001 in the cases where we use a single-head rotation classifier as R_1, and 0.003 in the multi-head case. The reason of this choice lies on the fact that, with a multi-head rotation classifier, we have to train 45 heads instead of a single one, so the training process needs to be speeded up. Moreover, we set a learning rate 10 times lower for the weights of the feature extractor and $C_{1/2}$. We also decrease the learning rate using a step learning rate scheduler, which reduces it by a factor of 10 after 90% of total epochs. All models are trained for 50 epochs for the step one and for 25 epochs for the step two. For models using the center loss, a centroid learning rate of 0.001 is used. The other hyperparameters are analyzed performing an ablation on them, details can be found on the section 4.4 and the results of the ablation are reported on appendix A.

Feature extractor E: ResNet-18 pre-trained on ImageNet. It is also shared between step 1 and step 2.

Classifier C_1: It is a simple linear classifier with a 45-dimensional output.

Classifier C_2: Like C_1 but with a 46-dimensional output to consider also the unknown class. Furthermore, it leverages the learning already performed on C_1. It is worth noting that we rebalance the weight of the loss computed on samples belonging to the class unknown in the following way. After having separated the target into $\mathcal{D}_t^{knw}$ and $\mathcal{D}_t^{unk}$, we select the weight as $\frac{\bar{\mathcal{D}}_s}{|\mathcal{D}_t^{unk}|}$, where $\bar{\mathcal{D}}_s$ is the average number of samples contained in the source known classes. This ensures that we correctly balance the unknown class during training in a way which is independent on the number of identified target unknown samples during the separation.

Classifier R_1, R_2: they are seen as discriminators composed by a linear layer ($1024 \rightarrow 256$), followed by a batch normalization layer and a leaky relu activation. The last layer is then a linear one for the actual classification ($256 \rightarrow 4$). In the case of multi-head rotation classifier, the discriminator is then composed of $|\mathcal{C}_s|$ different linear layers, both for the first and the last one. Moreover, the

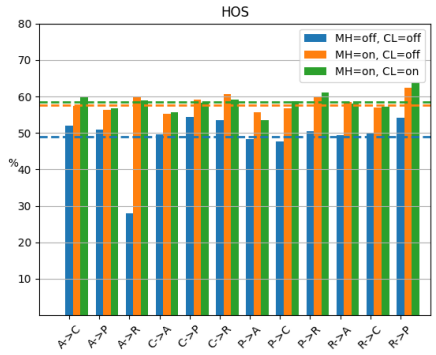

Figure 4. HOS values of the different configurations over different domain shifts.

256-dimensional output are the features used by the center loss for inferring class centroids. R_2 is always a single-head discriminator, even if using the multi-head variant.

4.3. Results

Figure 4 and table 4 summarize the performance, in terms of HOS values, obtained using different model configurations; in particular, we analyze the different combinations of having a multihead discriminator and the center loss additional contribution. The hyperparameters used for each model configuration are the best obtained from the ablation discussed in section 4.4. Moreover, we report the results obtained for all the 12 available domain shifts, as well as the average value to have an overall performance comparison. As can be seen, models with a multi-head discriminator obtain better AUROC values, resulting in a better separability and consequently in a better domain adaptation. Furthermore, we notice an additional improvement with the additional center loss contribution, showing that it effectively improves the discriminability of the hidden features in the learning process.

4.4. Ablation Study

We are now interested in conducting an ablation over the different hyperparameters for the different model configurations. In section 4.2 we have already discussed the architecture and the learning process, we thus proceed to sweep different values for all the hyperparameters to see how it affects the model performance in terms of HOS values. Given a specific configuration of hyperparameters values, we run the model for two domain shifts, Art $\rightarrow$ Clipart and Product $\rightarrow$ Clipart, and then we average the HOS values to obtain a more reliable estimate. We want to stress that all the hyperparameters might be highly correlated; therefore, a grid-search approach would be necessary. However, we decide to keep a sequential ablation study, analyzing the effectiveness of each hyperparameter in a specific order: α_1, λ, $\tilde{\mathcal{N}}$, α_2. Once the ablation for a hyperparameter is finished, we fix the best value to analyze the subsequent hyperparameters. Image 5 summarizes the results of the ablation, while the tables containing the values can be found in appendix A. First, we can observe that the performance of each hyperparameter follows a very similar trend indipendently from the chosen model configuration. An exception is the selection of a good threshold $\tilde{\mathcal{N}}$ to perform the target separation. We can see that for the configuration with a single-head and without the center loss contribution, it is required a larger

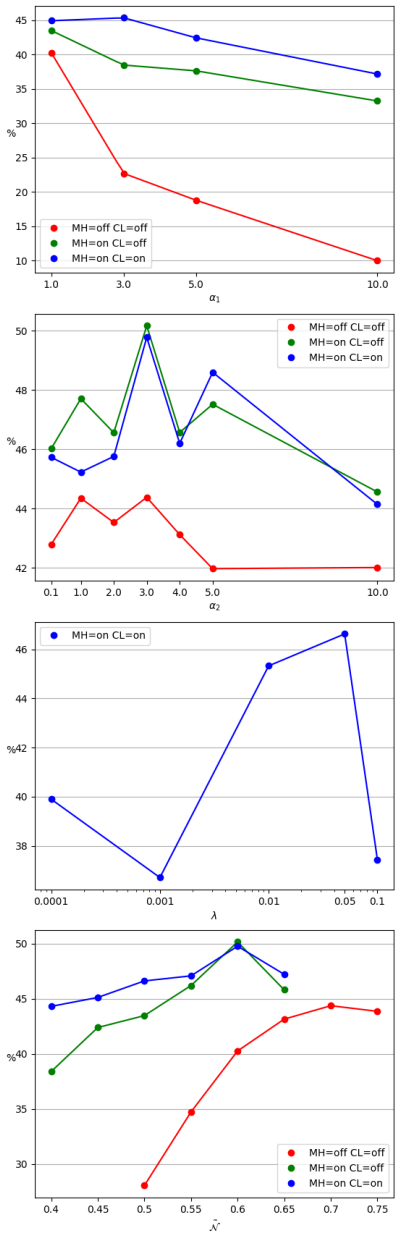

Figure 5. Ablation HOS values for different hyperparameters and model configurations. For the full details see appendix A.

value. We hypothesize that this happens as the single-head is not adequate to model the rotation of a large number of semantic classes. Infact, the model tends to give higher and less discriminant normality scores, achieving a worse AUROC value and performing a bad separation.

Smaller values of α_1 and α_2 are preferred as they avoid a too fast training convergence for the rotation classifier w.r.t the object classifier. Moreover, we notice that picking the right value for α_2 provides a good improvement. As the center loss contribution is directly summed to the rotation classifier loss term, we need to pick sufficiently small values of λ to avoid that it becomes too preponderant, making impossible the learning of the rotation task.

To sum up, we can state that different values of hyperparameters provide an unneglectable performance improvement, making the choice of the correct values important for the method effectiveness.

5. Future work and conclusions

The achieved results show that self-supervision techniques can be applied effectively in domain adaptation tasks and open-set classification tasks. We propose a method which leverages the self-supervision power to address the two challenges exploiting the relative rotation prediction task, by first separating the target domain into a known and unknown part, and then by performing the domain alignment. Moreover, we propose a new metric to evaluate OSDA systems, HOS, which is the harmonic mean between the accuracy on the target known samples and the accuracy on rejecting the unknown ones.

Some critical points still remain in our method of study and in our proposed solution. One point is the hyperparameters ablation method for different models, as small variations can cause huge differences in results due to the sequential optimization of hyperparameters. Furthermore, having models learn different tasks simultaneosly can lead to instability, so using slower learning models at the expense of longer training times can improve results. A huge criticality is represented by the normality score function, as simply picking the rotation prediction confidence (maximum predicted probability) does not provide a very discriminant score between images belonging to a known class and images of unknown classes. A better normality score function should take into considerations other factors to result in a more discriminant score. It is worth noting that the separation goodness influences the quality of the input material for the second step, resulting to be an extremely critical part for the effectiveness of the method. Lastly, the self-supervision task effectiveness also depends on the model architecture, as, for example, deeper architectures can result in a better or worse learning of domain-independent features, as underlined in [17].

A. Tables

This section contains the tables 1, 2 and 3 reporting the values of hyperparameters analyzed during the ablation, while the table 4 contains the values of the final models using the best hyperparameters, for each domain shift.

OFF-OFF			
α_1	$\tilde{\mathcal{N}}$	α_2	HOS$_{mean}$
Picking α_1			
1.0	0.6	3.0	**40.27%**
3.0			22.69%
5.0			18.78%
10.0			9.99%
Picking $\tilde{\mathcal{N}}$			
1.0	0.50	3.0	28.05%
	0.55		34.73%
	0.60		40.27%
	0.65		43.16%
	0.70		**44.38%**
	0.75		43.87%
Picking α_2			
1.0	0.7	0.1	42.79%
		1.0	44.35%
		2.0	43.53%
		3.0	**44.38%**
		4.0	43.12%
		5.0	41.97%
		10.0	42.01%

Table 1. Hyperparametr optimization process for configuration Multi-Head Off and Center-Loss Off

ON-OFF			
α_1	$\tilde{\mathcal{N}}$	α_2	HOS_{mean}
Picking α_1			
1.0	0.6	3.0	**43.48%**
3.0			38.45%
5.0			37.61%
10.0			33.23%
Picking $\tilde{\mathcal{N}}$			
1.0	0.4	3.0	38.40%
	0.45		42.40%
	0.5		43.48%
	0.55		46.21%
	0.6		**50.17%**
	0.65		45.82%
Picking α_2			
1.0	0.7	0.1	46.02%
		1.0	47.71%
		2.0	46.56%
		3.0	**50.17%**
		4.0	46.57%
		5.0	47.52%
		10.0	44.56%

Table 2. Hyperparameters optimization process for configuration Multi-Head On and Center-Loss Off

ON-ON				
α_1	λ	$\tilde{\mathcal{N}}$	α_2	HOS_{mean}
Picking α_1				
1.0	0.01	0.05	3.0	44.90%
3.0				**45.33%**
5.0				42.41%
10.0				37.15%
Picking λ				
3	0.0001	0.5	3.0	39.90%
	0.001			36.71%
	0.01			45.33%
	0.05			**46.63%**
	0.1			37.44%
Picking $\tilde{\mathcal{N}}$				
3	0.05	0.4	3.0	44.32%
		0.45		45.13%
		0.5		46.63%
		0.55		47.09%
		0.6		**49.78%**
		0.65		47.23%
Picking α_2				
3	0.05	0.6	0.1	45.73%
			1.0	45.23%
			2.0	45.76%
			3.0	**49.78%**
			4.0	46.20%
			5.0	48.60%
			10.0	44.14%

Table 3. Hyperparameters optimization process for configuration Multi-Head On and Center-Loss On

Results												
Shift	Single-Head, No Center Loss				Multi-Head, No Center Loss				Multi-Head, Center Loss			
	AUROC	HOS	UNK	OS*	AUROC	HOS	UNK	OS*	AUROC	HOS	UNK	OS*
A→C	52.07	44.76	58.65	36.19	57.46	46.64	62.95	37.04	59.77	52.68	62.73	45.40
A→P	50.84	51.27	57.56	46.23	56.31	54.13	60.42	49.03	56.86	52.46	63.35	44.77
A→R	27.94	68.13	17.58	52.25	59.88	56.60	53.18	60.65	58.86	61.67	66.82	57.26
C→A	49.52	41.74	43.10	40.47	55.28	42.63	51.57	36.33	55.76	46.84	60.19	38.35
C→P	54.42	46.99	41.51	54.15	59.13	51.88	60.03	45.69	58.35	52.69	62.19	45.72
C→R	53.43	46.18	38.64	57.39	60.75	56.00	61.29	51.63	59.15	55.97	63.18	50.25
P→A	48.32	37.07	33.70	41.20	55.62	41.72	48.90	36.39	53.50	40.97	49.53	34.94
P→C	47.76	43.99	48.42	40.31	56.77	46.10	55.11	39.65	58.64	46.81	60.95	37.99
P→R	50.37	34.14	22.88	67.27	59.68	48.41	40.91	59.30	61.10	56.22	56.89	55.58
R→A	49.39	25.08	15.99	58.19	58.47	45.00	39.50	52.43	57.98	48.92	45.61	52.77
R→C	49.80	46.70	48.27	45.23	57.02	49.00	42.82	57.34	57.09	48.69	61.11	40.47
R→P	54.10	23.37	13.89	73.69	62.33	50.31	39.51	69.27	63.96	59.64	54.40	66.02
AVG	49.00	42.45	36.68	51.05	58.23	49.03	51.35	49.56	**58.42**	**51.96**	58.91	47.46

Table 4. Final results for the three configurations on all domain shifts. All values are percentages.

References

[1] Abhijit Bendale and Terrance Boult. Towards open set deep networks, 2015. 2, 3

[2] Liron Bergman and Yedid Hoshen. Classification-based anomaly detection for general data, 2020. 2

[3] Konstantinos Bousmalis, Nathan Silberman, David Dohan, Dumitru Erhan, and Dilip Krishnan. Unsupervised pixel-level domain adaptation with generative adversarial networks, 2017. 2

[4] Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan. Domain separation networks, 2016. 2

[5] Silvia Bucci, Mohammad Reza Loghmani, and Tatiana Tommasi. On the effectiveness of image rotation for open set domain adaptation, 2020. 10

[6] Pau Panareda Busto and Juergen Gall. Open set domain adaptation. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 754–763, 2017. 2

[7] Fabio Maria Carlucci, Antonio D'Innocente, Silvia Bucci, Barbara Caputo, and Tatiana Tommasi. Domain generalization by solving jigsaw puzzles, 2019. 2

[8] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation, 2017. 2

[9] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations, 2020. 2

[10] Akshay Raj Dhamija, Touqeer Ahmad, Jonathan Schwan, Mohsen Jafarzadeh, Chunchun Li, and Terrance E. Boult. Self-supervised features improve open-world learning, 2021. 2

[11] Alexey Dosovitskiy, Philipp Fischer, Jost Tobias Springenberg, Martin Riedmiller, and Thomas Brox. Discriminative unsupervised feature learning with exemplar convolutional neural networks, 2015. 2

[12] Qianyu Feng, Guoliang Kang, Hehe Fan, and Yi Yang. Attract or distract: Exploit the margin of open set, 2019. 2

[13] Zeyu Feng, Chang Xu, and Dacheng Tao. Self-supervised representation learning from multi-domain data, 10 2019. 2

[14] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation, 2015. 2

[15] ZongYuan Ge, Sergey Demyanov, Zetao Chen, and Rahil Garnavi. Generative openmax for multi-class open set classification, 2017. 2

[16] Chuanxing Geng, Sheng-Jun Huang, and Songcan Chen. Recent advances in open set recognition: A survey, 2021. 1

[17] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations, 2018. 2, 3, 7

[18] Izhak Golan and Ran El-Yaniv. Deep anomaly detection using geometric transformations, 2018. 2

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. 5

[20] Matthias Hein, Maksym Andriushchenko, and Julian Bitterwolf. Why relu networks yield high-confidence predictions far away from the training data and how to mitigate the problem, 2019. 2

[21] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei A. Efros, and Trevor Darrell. Cycada: Cycle-consistent adversarial domain adaptation, 2017. 2

[22] Jingyun Jia and Philip K. Chan. Self-supervised detransformation autoencoder for representation learning in open set recognition, 2021. 2

[23] Mengmeng Jing, Jingjing Li, Lei Zhu, Zhengming Ding, Ke Lu, and Yang Yang. Balanced open set domain adaptation via centroid alignment, 2021. 3

[24] Shu Kong and Deva Ramanan. Opengan: Open-set recognition via open data generation, 2021. 2

[25] Hong Liu, Zhangjie Cao, Mingsheng Long, Jianmin Wang, and Qiang Yang. Separate to adapt: Open set domain adaptation via progressive separation. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2922–2931, 2019. 2

[26] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael I. Jordan. Learning transferable features with deep adaptation networks, 2015. 2

[27] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles, 2017. 2

[28] Pramuditha Perera and Vishal M. Patel. Deep transfer learning for multiple class novelty detection, 2019. 2

[29] Kuniaki Saito, Donghyun Kim, Stan Sclaroff, and Kate Saenko. Universal domain adaptation through self supervision, 2020. 2

[30] Kuniaki Saito, Yoshitaka Ushiku, and Tatsuya Harada. Asymmetric tri-training for unsupervised domain adaptation, 2017. 2

[31] Kuniaki Saito, Shohei Yamamoto, Yoshitaka Ushiku, and Tatsuya Harada. Open set domain adaptation by backpropagation, 2018. 2

[32] Walter J. Scheirer, Anderson de Rezende Rocha, Archana Sapkota, and Terrance E. Boult. Toward open set recognition, 2013. 2

[33] Yu Sun, Eric Tzeng, Trevor Darrell, and Alexei A. Efros. Unsupervised domain adaptation through self-supervision, 2019. 2

[34] Yaniv Taigman, Adam Polyak, and Lior Wolf. Unsupervised cross-domain image generation, 2016. 2

[35] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation, 2017. 2

[36] Hemanth Venkateswara, Jose Eusebio, Shayok Chakraborty, and Sethuraman Panchanathan. Deep hashing network for unsupervised domain adaptation, 2017. 2, 5

[37] Yandong Wen, Kaipeng Zhang, Zhifeng Li, and Yu Qiao. A discriminative feature learning approach for deep face recognition, 2016. 4

[38] Jiaolong Xu, Liang Xiao, and Antonio M. Lopez. Self-supervised domain adaptation for computer vision tasks, 2019. 2

[39] Kaichao You, Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I. Jordan. Universal domain adaptation. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2715–2724, 2019. 2

[40] Xiangyu Yue, Zangwei Zheng, Shanghang Zhang, Yang Gao, Trevor Darrell, Kurt Keutzer, and Alberto Sangiovanni Vincentelli. Prototypical cross-domain self-supervised learning for few-shot unsupervised domain adaptation, 2021. 2

[41] Richard Zhang, Phillip Isola, and Alexei A. Efros. Colorful image colorization, 2016. 2

[42] Yang Zou, Zhiding Yu, B. V. K. Vijaya Kumar, and Jinsong Wang. Domain adaptation for semantic segmentation via class-balanced self-training, 2018. 2