Open-Set Domain Adaptation through Self-Supervision

Protopapa Andrea, Quarta Matteo, Ruggeri Giuseppe, Versace Alessandro
Politecnico di Torino
Italy
{s286302,s292477,s292459,292435}@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.

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. As underlined in [4], this is usually achieved by learning a transformation function which projects both the source and target domain data in a new space where we jointly maintain the underlying structure of the original data while reducing the domain gap by removing the domain-specific information.

Another big issue encountered in real world usages is the presence of additional anomaly semantic classes on the observed data. The problem comes under the name of *Open-Set Recognition (OSR)* [5], where we require not only to accurately classify the known seen classes, but also effectively deal with unknown ones, which would otherwise drastically weaken the robustness of the methods.

The jointly presence and accounting of the two described problems has been emerged 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 domain and the source domain, we will observe a negative adaptation due to the unwanted alignment between the data belonging to the anomaly 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 novelty classes, translating the problem into a *Closed-Set Domain Adaptation (CSDA)* one, and next do the alignment between the source and the target domain identified as known.

Common machine learning methods usually leverage huge manually annotated datasets to perform well on the given tasks. However, acquiring annotated material is usually very costly and difficult, moreover, relying on such data may not be scalable in large applications on the long

run. Thus, 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, by solving such task, the model can learn the underlying structure of the data, for instance high-level knowledge, correlations, and metadata embedded. This type of learning has been recently used for Domain Adaptation, learning robust cross-domain features and supporting generalization [3, 10], and also for some Open Set problems specialized in anomaly detection and discriminating anomalous data [2, 7].

The approach presented in this paper combines the power of the self-supervised learning with the standard supervised learning approach for semantic class recognition. A two-stage method is hence proposed, aiming to identify and isolate unknown class samples in the first stage, before reducing in the second stage the domain gap between the source domain and the known target domain to avoid negative transfer. This is done in both stages 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 either to 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 [8] exploiting 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

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 ($\mathcal{C}_r = \{0, 1, 2, 3\}$).

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 [6]. We will further discuss possible improvements and variants in **PLEASE COMPLETE**.

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 [9] 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.

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).

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 contributions to make clear the usage of α_2 and we recall that $\mathcal{L}_{C_2}$ is now computed using the new arranged dataset $\mathcal{D}_s^*$.

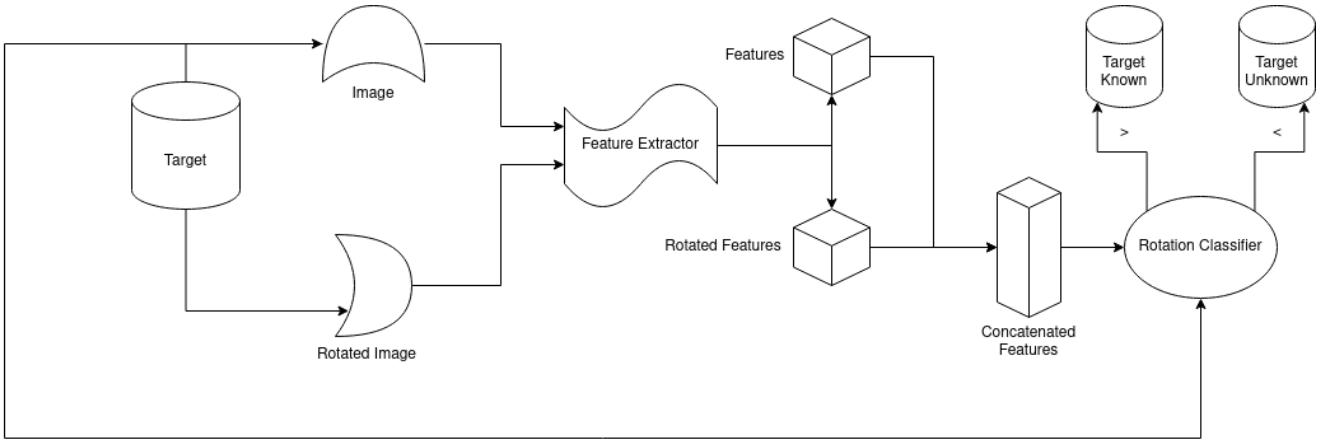

Figure 1. Schema representing the target images separation process

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

3.6. Performance Metrics

Evaluating model perfomance requires finding a balance between two values: *OS**, the fraction of correctly classified samples, and *UNK*, the fraction of correclty rejected samples. A model not confident enough to reject a sample could still achieve high *OS** values but near-zero *UNK*, while a model rejecting every sample as unkwnown will achieve perfect *UNK* and zero *OS**. To compare models we use the harmonic mean between *OS** and *UNK*, defined as $HOS = 2\frac{OS^* \times UNK}{OS^* + UNK}$. This is because this kind of mean tends gives a bigger weight to the smaller values, resulting in a more severe evaluation of models.

4. Experiments

4.1. Dataset

Our model is tested on the *Office-Home* dataset [8], which features 65 classes of images over four different domains: Art (A), Clipart (C), Product (P) and Real World (R). We set the first 45 classes to be known while the remaining 20 are unknown. For each experiment, we report both the best achieved HOS as well as the last reported HOS. As separation is crucial for the model effectiveness, we also report the computer AUROC score for the first part.

4.2. Results

Table 1 contains the results for all 12 **(ne abbiamo provati davvero 12???)** available domain shifts, as well as perfomance differences when using multi-head rotation classifiers and center loss. Further details are to be found in section 5. We can notice that multi-headed models are more capable of correctly discriminating between known and unknown samples, ultimately resulting in better domain alignment. A further advantage is given when combined with center loss. AUROC values are computed at the end of the separation stage, while HOS values are sampled at regular times on the entire original target dataset.

5. Implementation Details

5.1. Model Parameters

All models were run using stochastic gradient descent with batch size 32, a weight decay of 0.0005 and momentum of 0.9. For single-headed models, a base learning rate of 0.001 is used, while for multi-headed models it is set to 0.003. This is because only one head of the heads is trained with each samples, so the process needs to be sped up. Rotation classifier use a learning rate set to a tenth of the base learning rate. A step learning rate scheduler is used, reducing it by a factor of 10 after 90% of epochs. All models are run for 50 epochs for the first step described in 4 and for 25 epochs for the second step described in 3.5. For model using the center loss a centroid learning rate of 0.5 is used. All other parameters are configuration-sepcific and are reported in table 2. "No Rotation" model is the same as a single-head without center loss model with $\alpha_2 = 0$.

Extractor E: The extractor employed is a ResNet-34**[Inserire cit a ResNet] è pretrained??? ma 34 o 18???** for all classifiers.

Classifier C_1, C_2: C_1 is composed of a 45 dimensional output. All unknown samples are mapped as belonging to class 45. C_1 is used as a starting point for C_2. As no unknowns are present on the source dataset (they are removed), the weight of unknwon samples in the second step is set to **spiegare come è bilanciato** while for all other classes it is set to 1. A single fully connected layer $512 \rightarrow 45$ with batch-norm is used.

Single-Head, CE Loss				
Source	Target	AUROC	HOS	HOS*
S	T	50%	30%	30%
S	T	50%	30%	30%
S	T	50%	30%	30%
Multi-Head, CE Loss				
Source	Target	AUROC	HOS	HOS*
S	T	50%	30%	30%
S	T	50%	30%	30%
S	T	50%	30%	30%
Single-Head, CE+C Loss				
Source	Target	AUROC	HOS	HOS*
S	T	50%	30%	30%
S	T	50%	30%	30%
S	T	50%	30%	30%
Multi-Head, CE+C Loss				
Source	Target	AUROC	HOS	HOS*
S	T	50%	30%	30%
S	T	50%	30%	30%
S	T	50%	30%	30%
No Rotation				
Source	Target	AUROC	HOS	HOS_{Best}
S	T	50%	30%	30%
S	T	50%	30%	30%
S	T	50%	30%	30%

Table 1. Results for domain alignment. AUROC is the area under the ROC. HOS is latest achieved HOS, HOS* the best achieved through all epochs.

MH	CL	α_1	α_2	λ_1	$\tilde{\mathcal{N}}$
Off	Off	0.0	0.0	0.0	0.0
On	Off	0.0	0.0	0.0	0.0
Off	On	0.0	0.0	0.0	0.0
On	On	0.0	0.0	0.0	0.0

Table 2. Model Parameters

Classifier R_1, R_2: R_1 and R_2 are composed by a $1024 \rightarrow 256$ and a $256 \rightarrow 4$ fully connected layers, named *Discriminators*. The 256-dimensional output are the features used by the center loss for inferring class centers. If multi-head is used, each head is a separate discriminator. R_2 is always a single-head classifier, even if using a multi-head rotation classifier.

5.2. Ablation Study

As shown in table 1 different architectures give different results, and as shown in table 2 each one requires a different set of parameters. To choose optimal parameters a sequential approach is followed, by optimizing one parameter at the time. The order of optimization followed is: α_1, λ_1, $\tilde{\mathcal{N}}$, α_2. Models not using center loss just skip the λ_1 optimization. Ablation study were run on two domain shifts, Art $\rightarrow$ Clipart and Clipart $\rightarrow$ Product. In table **??** we report the steps followed. All other settings follow what is reported in 5.1. The result here obtained are the same reported in table 2.

MH OFF CL OFF				
α_1	$\tilde{\mathcal{N}}$	α_2	AUROC	HOS_{mean}
Picking α_1				
1.0	0.5	3.0	0.5	30%
3.0			0.5	30%
5.0			0.5	30%
10.0			0.5	30%
Picking $\tilde{\mathcal{N}}$				
1.0	0.40	3.0	0.5	50%
	0.50		0.5	50%
	0.55		0.5	50%
	0.60		0.5	50%
Picking α_2				
1.0	0.6	1.0	0.5	50%
		3.0	0.5	50%
		5.0	0.5	50%
		10.0	0.5	50%
Final Parameters				
1.0	0.6	3.0	0.5	50%

Table 3. Ablation performed for OFF-OFF configuration

6. Future Work

The results show that self-supervision technique can help in domain adaptation task and open-set classification tasks. A few critical points still remain on our method of study and proposed solution. The most important one is probably parameter choosing for different models, as we have seen that variations cause huge differences in results and sequential optimization of parameter is a sub-optimal heuristic. Furthermore, having the model to learn two different tasks at once, it could be useful to use a slower learning model at the expense of longer training times.

References

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

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

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

[4] Abolfazl Farahani, Sahar Voghoei, Khaled Rasheed, and Hamid R. Arabnia. A brief review of domain adaptation, 2020. 1

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

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

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

[8] Hemanth Venkateswara, Jose Eusebio, Shayok Chakraborty, and Sethuraman Panchanathan. Deep hashing network for unsupervised domain adaptation, 2017. 2, 4

[9] Yandong Wen, Kaipeng Zhang, Zhifeng Li, and Yu Qiao. A discriminative feature learning approach for deep face recognition, 2016. 3

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