Open-Set Domain Adaptation through Self-Supervision

Protopapa Andrea, Quarta Matteo, Ruggeri Giuseppe, Versace Alessandro
Politecnico di Torino
Italy
{s286302,s292477,s292459,s292477}@studenti.polito.it <- Riordinare

Abstract

Performance loss over domain change is a common problem in machine learning applications. Solving the issue usually relies on adapting the model to the new domain by having at disposal large and labeled datasets, either rare or costly to acquire. In this paper we propose a new approach, adapting models to the new domains while not needing relying on labeled data thanks to a self-supervision. Experiments conducted on the Office-Home dataset show interesting results and method effectiveness.

1. Introduction

Classical machine learning in the past years has made some oversimplified assumptions actually detached from the usage of artificial intelligence systems in everyday real world and the problems they come with.

The first assumption is that training and test sets come from the same distributions: a model trained on labeled data is expected to perform as well as on the test data. However, this assumption not always holds in real-world applications, where naively applying the trained model on a new dataset may cause degradation in the performance. To solve this problem Domain Adaptation is widely used, where the goal is to train a neural network on a source dataset for which labels are available and search for good performance on a target dataset, which is related to but significantly different from the source dataset, and whose label or annotation is not available. Generally this is done by by minimizing the difference between domain distributions and enforcing the recognition of domain invariant patterns in both domains. As underlined in [4], this is usually achieved by mapping source and target data by learning transformations for extracting such features and minimizing the gap between domains in the new representation space in an optimization procedure, while preserving the underlying structure of the original data.

Secondly, in real-world classification tasks it is usually difficult to collect training samples to exhaust all classes when training a model. A more realistic scenario is open set recognition (OSR) [5], where incomplete knowledge of the world exists at training time and unknown classes can be injected during testing, requiring classifiers to not only accurately classify the seen classes but also effectively deal with unseen ones, which would otherwise drastically weaken the robustness of the methods. On the contrary this system should reject unknown classes at test time and separate the known and unknown samples. As underlined by [10], existing open-set classifiers rely on deep networks trained in a supervised fashion on known classes in the training set; this causes specialization of learned representations to known classes and makes it hard to distinguish the unknowns from the knowns.

To solve these significant issues our method is focused on a self-supervised task. Self-supervised learning is an unsupervised learning technique where the supervised task is created out of the unlabeled input data. This task could be as simple as predicting the lower half of an image being given the upper half of the same. Supervised learning requires both labeled and high quality data, usually very expensive, whereas unlabeled data is often readily available in abundance. The fundamental idea behind self-supervised learning 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 in the data. This type of learning was recently used for Domain Adaptation, learning robust cross-domain features and supporting generalization [3,9], and also for some Open Set problems specialized in anomaly detection and discriminating anomalous data [2, 6].

The approach presented in this paper brings these topics together in the so called Open-Set Domain Adaptation (OSDA) problem. 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 model, 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.

The method was evaluated on the Office-Home benchmark [7] with a specific OSDA metric.

ADD HERE RESULTS AND A BRIEF OF CONCLUSIVE IDEAS (ALSO POSSIBLE FUTURE WORKS)!!

2. Related Work

3. Method

3.1. Problem Formulation

Our starting point is the source dataset, defined as $\mathcal{D}_s = \{(\mathbf{x}_i^s, y_i^s)\}_{i=1}^{N_S} \sim p_s$, where each element $\mathbf{x}_i^s$ belonging to class y_i^s is a sample from the source domain S. This dataset has a target counterpart, $\mathcal{D}_t = \{\mathbf{x}_i^t\}_{i=1}^{N_t} \sim p_t$ which is unlabeled instead. In OSDA we have that $p_s \neq p_t$. The source dataset $\mathcal{D}_s$ is associated with a set of known classes $\mathcal{C}_s$, which can also be found in the target dataset $\mathcal{D}_t$ but is supposedly smaller. In other words, $|\mathcal{C}_s| < |\mathcal{C}_t|$ and $\mathcal{C}_s \subset \mathcal{C}_t$. In domain adaptation, we further have that the target distribution of the known source classes differs from the target distribution of known classes, so $p_t^s \neq p_s$, A metric to measure how different two domains are is the openness between source and target domain [1], defined as $\mathbb{O} = 1 - \frac{\mathcal{C}_s}{\mathcal{C}_t}$. When $\mathbb{O} > 0$, we're dealing with an OSDA problem.

3.2. Approach

The task is split in two separate stages. First, we have to train the model to separate between the known classes ($\mathcal{C}_s$) and the unknown classes ($\mathcal{C}_{t \setminus s}$) in a reliable way. This is achieved via the self-supervised task, training the model to both recognize the class of a sample and its relative orientation. Secondly, we train the model on a union of both source and target datasets, to reject samples when they are not part of the known classes $\mathcal{C}_s$.

3.3. Rotation Recognition

We denote with $rot(\mathbf{x}, k)$ the rotation of the sample image $\mathbf{x}$ by $k \times 90$ degrees clockwise. This is the self-supervised part of our proposed model as rotation indices $k \in [0, 3]$ can be randomly generated and then predicted. Relative rotation, meaning that both original and rotated features are supplied to the model, is preferred over just using rotated features as some objects might not have an absolute correct orientation. Alternatively, instead of having the model predict the rotation of a sample for any class using the same classifier, we also try using a different head for each one of the known classes $\mathcal{C}_s$.

3.4. Step I: Sample Separation

Separating samples requires iterating over $\tilde{\mathcal{D}}_s = \{(\mathbf{x}_i^s, \tilde{\mathbf{x}}_i^s, z_i^s)\}_{i=1}^{N_s}$, where $\tilde{\mathbf{x}}_i^s$ is the $z_i^s \times 90$ degrees rotated version of $\mathbf{x}_i^s$. The network is composed by an extractor E and two classifiers, R_1 and C_1. C_1 is the object classifier, assigning image labes $\tilde{y}_i^s \in \{0, \dots, |\mathcal{C}_s|\}$, while R_1 is the relative rotation classifier, assigning rotation indices $\tilde{z}_i^s \in [0, 3]$. $\tilde{\mathbf{z}}_i^s$ and $\tilde{\mathbf{y}}_i^s$ are the vector representations of $\tilde{z}_i^s$ and $\tilde{y}_i^s$. The multi-head rotation classifier uses $|\mathcal{C}_s|$ heads for the rotation task. In this case, which head to use during separation is up to the object classifier C_1. Object predictions are made as $\tilde{\mathbf{x}}_i^s = softmax(E(\mathbf{x}_i^s))$, while relative rotation predictions are computed as $\tilde{\mathbf{z}}_s = softmax(R_1([E(\mathbf{x}^s), E(\tilde{\mathbf{x}}^s)]))$. If a multi-head rotation classifier is used, the $j = y_i^s$ head is used as $\tilde{\mathbf{z}}_s = softmax(R_{1,j}[E(\mathbf{x}^s), E(\tilde{\mathbf{x}}^s)])$. The model is trained to minimizing the objective function $\mathcal{L}_1 = \mathcal{L}_{C_1} + \mathcal{L}_{R_1}$. This is the sum of two cross-entropy loss functions:

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

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

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 [8] term:

$$\mathcal{L}_{R_1}^* = -\alpha_1 \sum_{i \in \mathcal{D}_s} \mathbf{y}_i^s \log \tilde{\mathbf{z}}_i^s + \lambda_1 ||\mathbf{v}_i^s - \gamma(\mathbf{z}_i^s)||_2^2 \tag{3}$$

Here $\mathbf{v_i}$ is the penultimate layer of the rotation classifier, called features, $\gamma(\mathbf{z}_i)$ is the center of the features associated to class i, $|| \cdot ||_2^2$ is the l-2 norm and λ_1 is the weight associated with the center loss term.

When training is completed, we can start separating samples. To do so, normality scores $\mathcal{N}(\cdot)$ are used, defined as the maximum prediction of the rotation classifier: $\mathcal{N}(\tilde{\mathbf{x}}_i) = \max(\tilde{\mathbf{z}}_i)$. To predict wheter a sample belongs to the known samples of the target domain $\mathcal{D}_t^{knw}$ or the unknown ones $\mathcal{D}_t^{unk}$ requires picking a threshold $\tilde{\mathcal{N}}$. The separation is then done as:

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

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 by choosing j as $j = \arg\max_j \tilde{\mathbf{y}}_i^t$.

3.5. Step II: Domain Alignment

In this phase, we build two "new" datasets. The first one is $\mathcal{D}_s^*$, composed as $\mathcal{D}_s \cup \mathcal{D}_t^{unk}$. Since $\mathcal{D}_t$ is an unlabeled dataset but $\mathcal{D}_s^*$ is a labeled one, samples in $\mathcal{D}_t^{unk}$ are labeled as unknowns. The second one is $\mathcal{D}_t^{knw}$. Here we also use two new classifiers, C_2 and R_2, similarly to how they were used in the first phase but with C_2 predicting $|\mathcal{C}_s| + 1$ classes to accomodate unknown predictions. In this step R_2 is always single-headed and has its own weight α_2. The objective function is the same described in equations 1 and 2. The main difference here is that C_2 is actually trained to recognize unknown classes, on both domains.

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 [7], 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 5.2. 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.

Single-Head, CE Loss				
Source	Target	AUROC	HOS	HOS$_{Best}$
S	T	50%	30%	30%
S	T	50%	30%	30%
S	T	50%	30%	30%
Multi-Head, CE Loss				
Source	Target	AUROC	HOS	HOS$_{Best}$
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$_{Best}$
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$_{Best}$
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. Test Caption

5. Implementation Details

5.1. Model Parameters

All models were run using a Stochastic Gradient Descent with batch size 32, a weight decay of 0.0005 and momentum set to 0.9. For single-headed models, the base learning rate is set to 0.001 and for multi-headed models it is set to 0.003. This is because multi-headed models take a longer time to converge. All rotation classifier have the learning rate set to a tenth of the base learning rate. Learning rates are reduced 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. Other parameters are configuration-sepcific and reported in table 2. When considering a "No Rotation" model, it is meant that $\alpha_2 = 0$ is used.

5.2. Ablation Study

As shown by table 1, different architectures greatly improve performance, and as shown by table 2 each one has its set of optimal parameters. To choose the best parameters, a sequential approach is followed, by optimizing one parameter at the time. The order of optimization followed

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

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

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. 1

[3] Fabio Maria Carlucci, Antonio D'Innocente, Silvia Bucci, Barbara Caputo, and Tatiana Tommasi. Domain generalization by solving jigsaw puzzles, 2019. 1

[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] Izhak Golan and Ran El-Yaniv. Deep anomaly detection using geometric transformations. 2018. 1

[7] Hemanth Venkateswara, Jose Eusebio, Shayok Chakraborty, and Sethuraman Panchanathan. Deep hashing network for unsupervised domain adaptation, 2017. 2, 3

[8] Yandong Wen, Kaipeng Zhang, Zhifeng Li, and Yu Qiao. A discriminative feature learning approach for deep face recognition, 2016. 2

[9] Jiaolong Xu, Liang Xiao, and Antonio M. Lopez. Self-supervised domain adaptation for computer vision tasks. 2019. 1

[10] Ryota Yoshihashi, Wen Shao, Rei Kawakami, Shaodi You, Makoto Iida, and Takeshi Naemura. Classification-reconstruction learning for open-set recognition, 2019. 1

is: $\alpha_1, \lambda_1, \tilde{\mathcal{N}}, \alpha_2$. Models not using center loss skip the λ_1 optimization. Ablation study were run on two domain shift, Art $\rightarrow$ Clipart and Clipart $\rightarrow$ Art. In table **??** we report the steps followed, picking the parameter that gives the highest mean HOS between the two shifts. All other settings follow what is reported in 5.1. For compactness, here are the final results of the data otherwise reported from 3 on.

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.