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 have made some oversimplified assumptions actually detached from the usage of Artificial Intelligent systems in everyday real world and the problems they bring.

The firt assumption is that the training and test sets come from the same distributions. Therefore, a model learned from the labeled training data is expected to perform well on the test data. However, this assumption may not always hold in real-world applications where the training and the test data fall from different distributions and in this case naively applying the trained model on the new dataset may cause degradation in the performance. To solve this problem we can make use of Domain Adaptation, where our goal is to train a neural network on a source dataset for which labels are available, and test a good accuracy on the target dataset, which is related but significantly different from the source dataset, and whose label or annotation is not available. Generally it seeks to learn a model from a source labeled data that can be generalized to a target domain by minimizing the difference between domain distributions and enforcing feature extractor to extract similar features for source and target domain images. As underlined in [4], feature-based adaptation approaches aim to map the source data into the target data by learning a transformation that extracts invariant feature representation across domains, transforming the original features into a new feature space and then minimize 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 recognition/classification tasks it is usually difficult to collect training samples to exhaust all classes when training a recognizer or classifier. 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 submitted to an algorithm during testing, requiring the classifiers to not only accurately classify the seen classes but also effectively deal with unseen ones, which otherwise drastically weakens the robustness of the methods. On the contrary this system should reject unknown/unseen 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 manner on known classes in the training set; this causes specialization of learned representations to known classes and makes it hard to distinguish unknowns from knowns.

To solve these significant issues this method is focused on a self-supervised task. Self-supervised Learning is an unsupervised learning method where the supervised learning task is created out of the unlabelled input data. This task could be as simple as given the upper-half of the image, predict the lower-half of the same image. Supervised learning requires usually a lot of labelled data and getting good quality labelled data is an expensive and time-consuming task. On the other hand, the unlabelled data is readily available in abundance. The fundamental idea for self-supervised learning is to create some auxiliary pre-text task for the model from the input data itself such that while solving the auxiliary task, the model learns the underlying structure of the data (for instance the high-level knowledge, correlations, 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 for anomaly detection and discriminate normal and 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 shift between the source domain and the knwon target domain to avoid negative transfer. All 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 in the latter case rejected.

The method was evaluated on the Office-Home benchmark [**?**] 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 any 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. 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. Hence we have that $|\mathcal{C}_s| < |\mathcal{C}_t|$ and that $\mathcal{C}_s \subset \mathcal{C}_t$. In a setting of domain adaptation, we further have that $p_t^s \neq p_s$, the target distribution of the known source classes A metric for measuring how different two domain are is the openness betweeen source and target domain [1], defined as $\mathbb{O} = 1 - \dfrac{\mathcal{C}_s}{\mathcal{C}_t}$. When $\mathbb{O} > 0$, we're dealing with an OSDA problem.

3.2. Approach

To tackle the task, we chose to split it in two different steps. 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 enough way. This is achieved by using a semi-supervised task, by training to model to both recognize a sample class and its correct orientation. The second step is similar to a classic CSDA problem, where we train the model on a union of both source and target datasets.

3.3. Rotation Recognition

Let's denote with $rot(\mathbf{x}, i)$ the rotation of the sample image $\mathbf{x}$ by $i \times 90$ degrees clockwise. This is the self-supervised part of our proposed model as rotations $i \in [0, 3]$ can be randomly generated and then predicted. To avoid situations where the objective orientation of a sample would be a too complicated task, even for a human being, we also feed in input to the model the un-rotated image features. Alternatively, instead to have the model predict the rotation of a sample for any class, we also try using a different head for each one of the known classes $\mathcal{C}_s$, along with different loss functions.

3.4. Step I: Sample Separation

To separate samples we train the model on an enhanced version of $\mathcal{D}_s$, $\tilde{\mathcal{D}_s} = \{(\mathbf{x}_i^s, \tilde{\mathbf{x}}_i^s, z_i^s)\}_{i=1}^{N_s}$ where $\tilde{\mathbf{x}}_i^s$ is the rotated version of $\mathbf{x}_i^s$ and z_i^s is the rotation index associated to image i. The network is composed by an extractor E and two classifiers, R_1 and C_1 in its standard form. When using a multi-head rotation classifier, it is composed of $|\mathcal{C}_s| + 1$ heads for the rotation task, and an additional one for the classification task. When using a multi-head predictor, which head to use during separation is chosen by the object classifier C_1. The features of both original and rotated image are used to predict the rotation as $\tilde{\mathbf{z}}_s = softmax(R_1([E(\mathbf{x}^s), E(\tilde{\mathbf{x}}^s)]))$ in the single-head case and using the j-th head as $\tilde{\mathbf{z}}_s = softmax(R_{1,j}[E(\mathbf{x}^s), E(\tilde{\mathbf{x}}^s)])$ in the multi-head case. Classes are predicted only using un-rotated features as $\tilde{\mathbf{y}}^s = softmax(C_1(E(\mathbf{x}_i^s)))$. The model is training by minimizing an objective function defined as $\mathcal{L}_1 = \mathcal{L}_{C_1} + \mathcal{L}_{R_1}$. This is the sum of two cross-entropy loss functions as in:

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

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

Where α_1 is a weight associated to the rotation task. We also try using an extended rotation objective function $\mathcal{L}_{R_1}^*$ also implementing a center loss function [8]:

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

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

When training is completed, we can start separating samples. To do so, we get the normality score $\mathcal{N}(\cdot)$ which is defined as the maximum prediction of the rotation classifier, $\mathcal{N}(\tilde{\mathbf{x}}_i^s) = \max(\tilde{\mathbf{z}}_i^t)$. To tell wheter a sample belongs to the known samples of the target domain $\mathcal{D}_t^{knw}$ or the unknown one $\mathcal{D}_t^{unk}$ requires choosing a threshold $\tilde{\mathcal{N}}$. Then we can simply separate as:

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

When employing a multi-head architecture, we need to choose which one of the $|\mathcal{C}_s|$ heads to use for the classification task. Head $R_{1,j}$ is used by picking j as $j = \arg\max_j \tilde{\mathbf{y}}_i^t$.

3.5. Step II: Domain Alignment

In this phase, two ”new” datasets are used. The first one is $\mathcal{D}_s^*$, composed as $\mathcal{D}_s \cup \mathcal{D}_t^{unk}$. Note that since $\mathcal{D}_t$ is an unlabeled dataset but $\mathcal{D}_s^*$ is a labeled one, all samples in $\mathcal{D}_t^{unk}$ are simply labeled as unknowns. The second one is $\mathcal{D}_t^{knw}$ which is still an unlabeled dataset. Here we also introduce two new classifiers, C_2 and R_2. C_2 is the object classifier, but while C_1 had a $|\mathcal{C}_s|$-dimensional output, having to accomodate the new ”Unknown” class, it has a $|\mathcal{C}_s|+1$-dimensional output. R_2 instead is the rotation classifier, which in this step is always single-headed and has its own weight α_2. These two classifiers try to minimize the objective function described in equations 1 and 2. One of the main differences between the two steps is that while in the first one we used C_1 to regularize the rotation task, here we use R_2 to regularize the main classification task.

3.6. Performance Metrics

Evaluating model perfomance requires finding a balance between two values: *OS**, the share of correctly classified samples; *UNK*, the share of correclty rejected samples. A model never confident enough to reject a sample as unknown could still achieve high *OS** scores and near-zero *UNK*, while a model marking every given sample as unkwnown will achieve perfect *UNK* and zero *OS**. To compare models in a robust manner, we picked an harmonic mean between *OS** and *UNK*, defined as $HOS = 2\frac{OS^* \times UNK}{OS^* + UNK}$. This type of mean is more biased towards the lowest of the two scores, 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. After each epoch performed during the domain alignment phase, a validation run is performed on the entire original target dataset. For each experiment, we report both the best achieved HOS and the last reported HOS. As separation is crucial for the model effectiveness, we also report the computer AUROC score for the first part.

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

4.2. Results

Table 1 contains the results for all 12 (ne abbiamo provati davvero 12???) possible domain shifts, as well as perfomance differences when using multi-head rotation classifiers and center loss. Each mode has a specific configuration further discussed in 5.2.

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,

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	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

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 is: α_1, λ_1, $\tilde{\mathcal{N}}$, α_2. Models not using center loss skip the λ_1 optimization. Ablation study were run on two domain shift, Art → Clipart and Clipart → 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.

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. 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