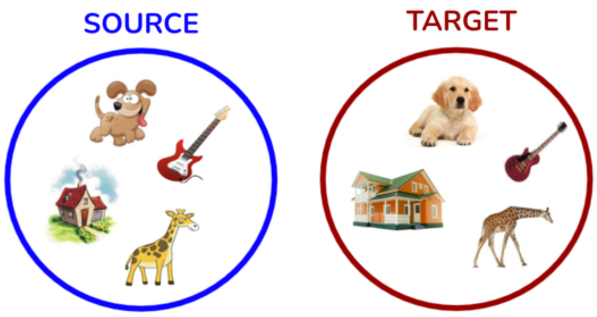
Open-Set Domain Adaptation through Self-Supervision

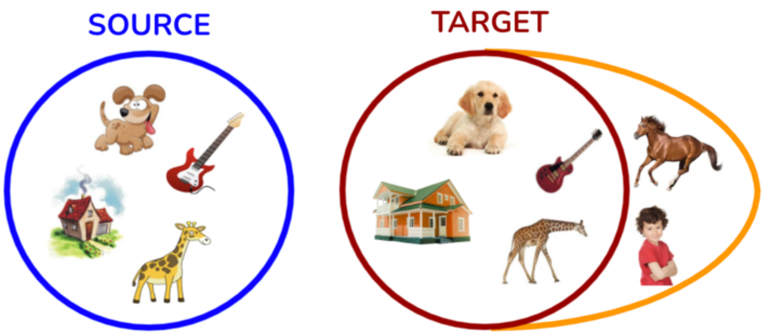
TA: Silvia Bucci
silvia.bucci@polito.it

Problem

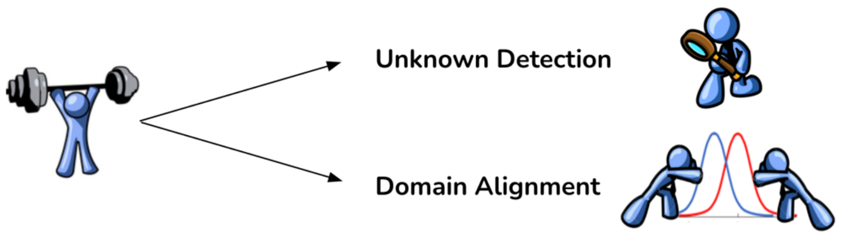
One of the most relevant problems that may affect any computer vision research area (object recognition, action recognition, semantic segmentation, etc.) is the discrepancy between the training data and the test data. The difference between the two distributions could lead the model to have poor performances at test time, causing several problems in real-world applications in which it's rare to know in advance the kind of data the model will have to deal with. From a technical point of view, the training set is named **Source Domain** and the test set is named **Target Domain**, the problem is formalized as **Domain Adaptation**: the source domain is a labeled dataset that the model can use during training, the target domain instead is provided unlabeled and only their images can be used for the adaptation of the model.



Since the target domain is unlabeled, we don't have any guarantee that it has the same set of categories as the source domain. A very realistic situation is that the source domain covers only a subset of the huge set of categories that a target domain could contain. This problem is formalized as **Open-Set Domain Adaptation.**



The model can learn only the categories of the source domain since it is the only labeled dataset. So, it's important to individuate the target samples from the unknown categories before making a surely wrong category prediction on them.
In this scenario the challenges are two: i) separate the target domain in known and unknown, ii) perform the alignment only between the source domain and the target-known domain.

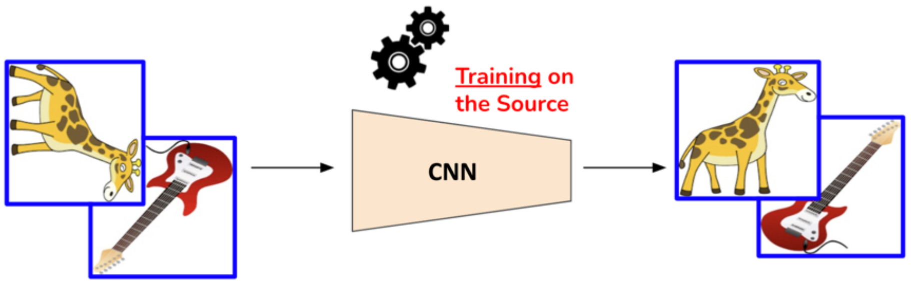


Proposed Solution

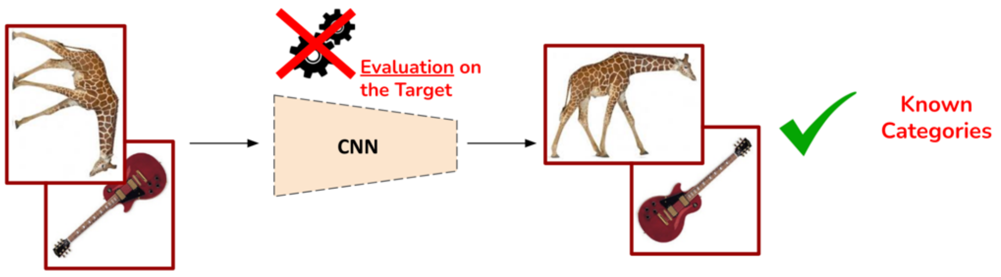
The domain-invariant regularities learned solving a **Self-Supervised** task can be a great help for both unknown detection and domain adaptation.

UNKNOWN DETECTION

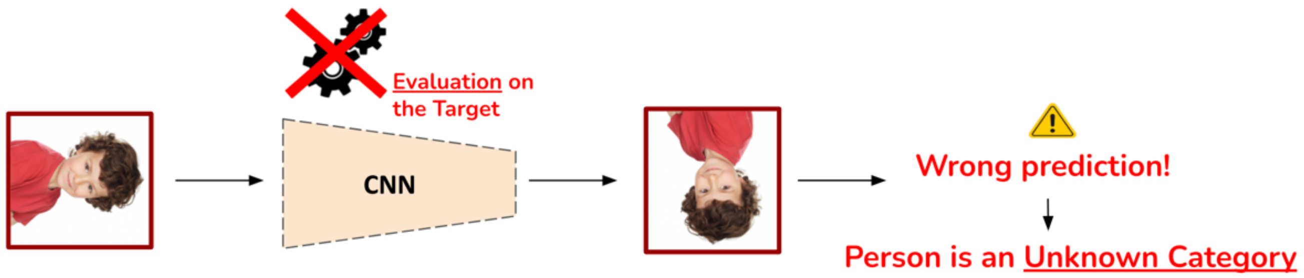
Considering the **Rotation Recognition** task:
the network is trained on the Source domain to predict the correct orientation of the objects.



Since the orientation of an object is domain-invariant, if the network trained on the source domain correctly predicts the orientation of a target image it means that the object is known.



If the network can't predict the orientation of a target image it is discarded as unknown.



DOMAIN ADAPTATION

Once that the target is divided into known and unknown, the adaptation step can be performed only between the source domain and the known part of the target domain. In this second step, it is used the rotation recognition task in a multi-task approach (together with the standard supervision) in order to reduce the domain gap between the two distributions still leveraging on the generalized knowledge acquired with the rotation recognition task.

Objectives

1. Read **[1]** to study the rotation recognition self-supervised task.
   Read **[2]** in which geometric transformations are exploited to do anomaly detection.
   Read JiGen **[3]** that proposes to use a self-supervised task to perform domain adaptation.
   Read ROS **[4]** in which [2] and [3] are used together to obtain a general approach to solve both unknown detection and domain adaptation using self-supervision.
2. Implement a simpler version of ROS following the instructions.
3. Implement at least one variation among those suggested.

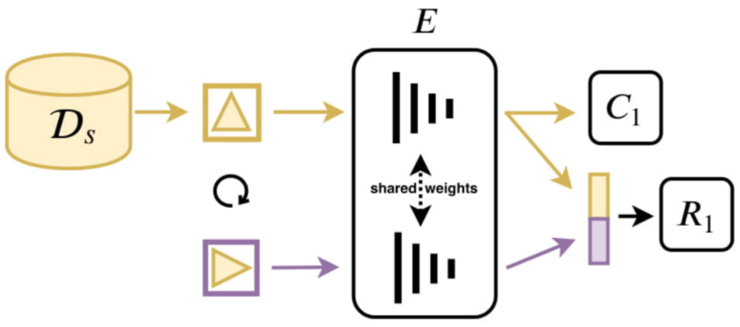
Mandatory

Starting from the GitHub repository [5] provided, first you have to carefully follow all the instructions in the *README.md* file in order to set the **Dataset** and the **Environment**. Then you can start editing the code.

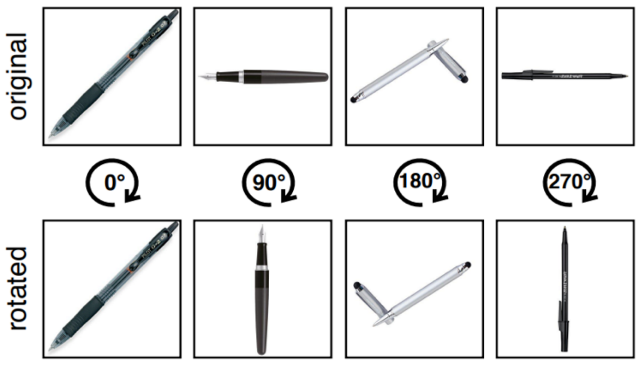
Implementation ---

Step1

In this step, you can use the source domain to train both an object classifier and the rotation recognition task.



You can implement the rotation recognition task as a **classification problem**. The angles of rotation are **0°, 90°, 180°, 270°** that correspond to labels **0, 1, 2, 3.** To predict the image orientation you can consider the concatenation of the features obtained with the not-rotated version of the mage and the rotated version of the same image. In this way, the network will learn the **relative orientation** of the object because sometimes could be some ambiguity in predicting the absolute orientation.



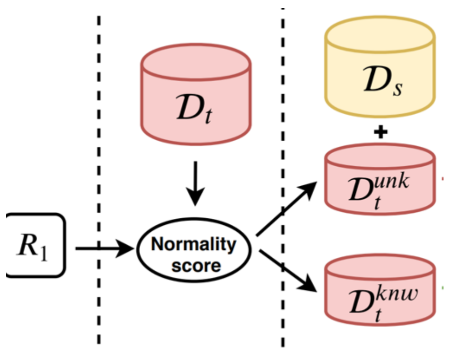
In this step, you will train two loss functions in a multi-task way: the loss function for the object recognition and the loss function for the rotation recognition. You can use a hyperparameter to control the contribution of the two losses in the total function:

$$L_{tot} = L_{cls} + \alpha_1 L_{rot}$$

Target Evaluation

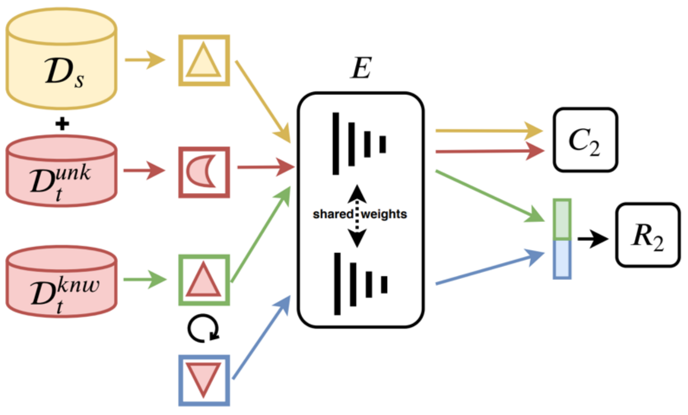
Once that you reach a well-trained rotation classifier, you can use it for the evaluation of the target domain. The idea is that, if the rotation classifier can predict the orientation of a target image it means that the image is from a known category. So for each target image, you will have a **normality score** that is the maximum value of the output of the rotation classifier (that is the **maximum prediction**). Once that you have a normality score for each target sample you can evaluate its precision through the **AUROC** metric [6]. A random AUROC is 0.5, so you have to obtain a value > 0.5 in order to know that your method is working.

Once that you have the vector of the normality scores and an AUROC > 0.5 you can do the known-unknown separation of the target. You have to set a **threshold** value, the samples with a normality score greater than the threshold will be considered known, the target samples with a normality score lower than the threshold will be considered as unknown.



Step2

The unknown part of the target will be used to train the unknown class in Step2, the known part instead will be used for the source-target adaptation.



Also in this case you have to implement a multi-task learning considering the rotation recognition task to perform domain adaptation between the source the target known samples.

$$L_{tot} = L_{cls} + \alpha_2 L_{rot}$$

Final Evaluation

After the last training step you have to evaluate the final model on the target domain according to three different metrics:

- **OS*** -> The accuracy of the object classifier in recognizing the known category
- **UNK** -> The accuracy of the object classifier in recognizing the unknown category
- **HOS** -> the harmonic mean between OS* and UNK

Hyperparameters Ablation

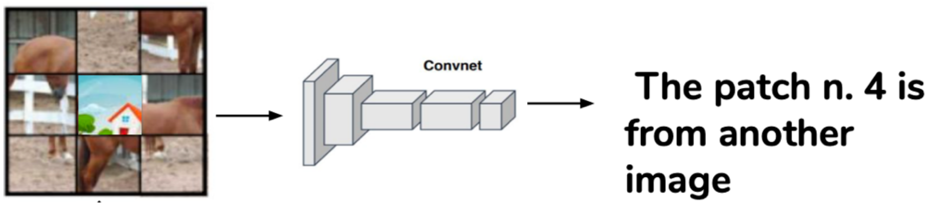
You have to do a study changing the value of α_1 and α_2 and the threshold for known/unknown separation in a range of possible values (for example, from 0.1 to 1) and see what happens to the performances of the model.

Variations

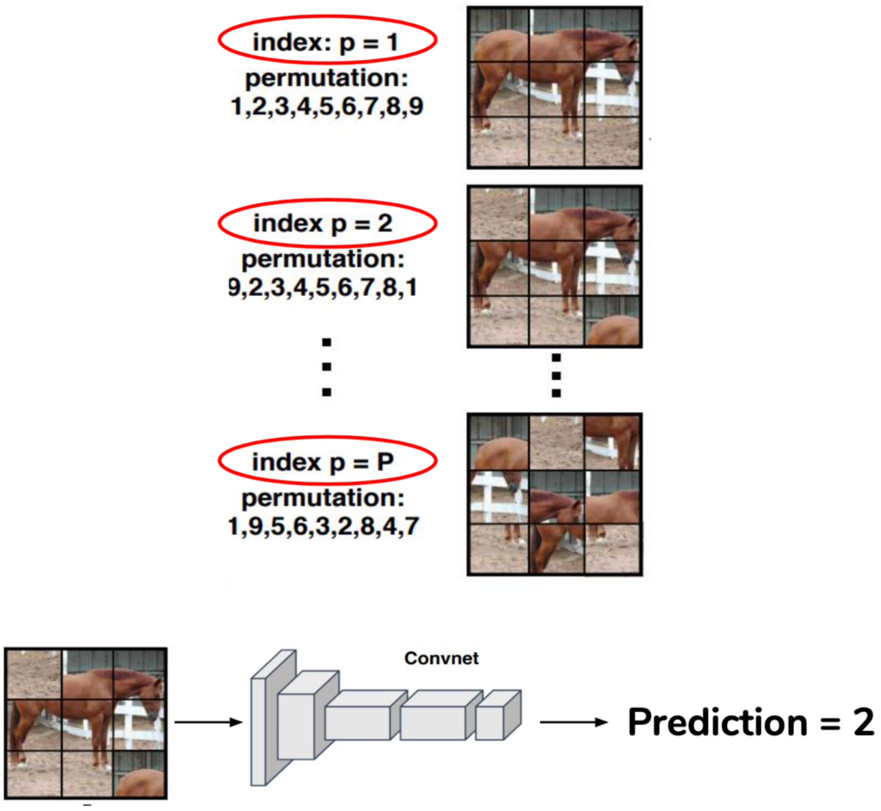
1. Implement at least two different Self-Supervised tasks

Implement at least two different self-supervised tasks instead of rotation recognition. Examples of self-supervised tasks are:

a) **Odd-One-Out**



b) **Jigsaw puzzle**


index: p = 1
permutation: 1,2,3,4,5,6,7,8,9
index p = 2
permutation: 9,2,3,4,5,6,7,8,1
index p = P
permutation: 1,9,5,6,3,2,8,4,7
Convnet
Prediction = 2

You can choose the proposed ones, but you can also propose a different one that intrigues you taking inspiration from [2] (as horizontal flipping, translations, etc.)

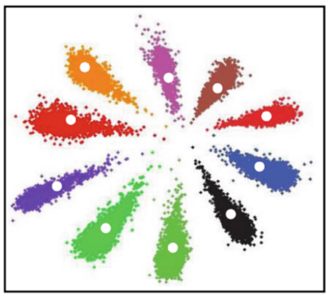
2. Implement a multi-head Rotation Classifier

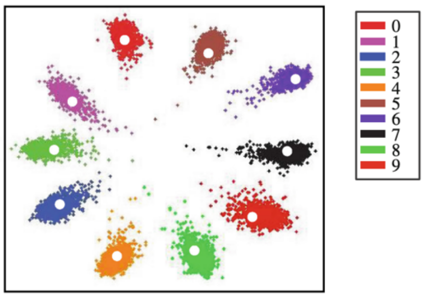
Until now we are using a single fully connected layer (single-head) for the rotation recognition task. A more precise way to perform unknown detection is to use a different fully connected layer for each known class (**multi-head**): you should have one head for the prediction of the orientation of samples from class 0, one head for the prediction of the orientation of samples from class 1, and so on.

This corresponds to the way in which this self-supervised task is implemented in ROS and it should be more effective.

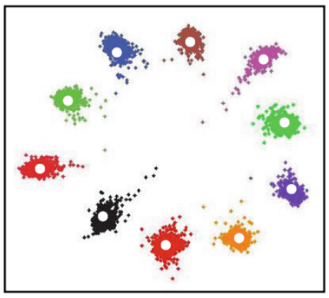
3. Add the Center-Loss

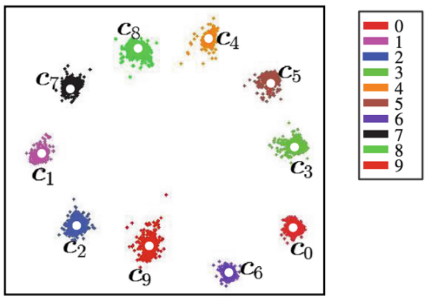
Study the center-loss [7] and try to apply it following this GitHub repository [8] in Step1. This loss should improve the rotation recognition task since simultaneously it learns a center for deep features of each class and penalizes the distances between the deep features and their corresponding class centers.
If you add this contribution in the total loss of Step1, you have to introduce a new hyperparameter and so you have to perform an ablation analysis also on it.


(a) $\lambda = 0.001$



(b) $\lambda = 0.01$


(c) $\lambda = 0.1$



(d) $\lambda = 1$

References:

[1] Komodakis, N., Spyros G.: Unsupervised representation learning by predicting image rotations. *In: ICLR* (2018)

[2] Golan, I., El-Yaniv, R.: Deep anomaly detection using geometric transformations. In: NeurIPS (2018)

[3] Carlucci, F.M., D'Innocente, A., Bucci, S., Caputo, B., Tommasi, T.: Domain generalization by solving jigsaw puzzles. In: CVPR (2019)

[4] Bucci, S., Loghmani, M. R., Tommasi, T.: On the effectiveness of image rotation for open set domain adaptation. In: ECCV (2020)

[5] https://github.com/silvia1993/ROS_AMLProject/tree/master

[6] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_auc_score.html

[7] Wen, Yandong, et al. "A discriminative feature learning approach for deep face recognition." *European conference on computer vision*. Springer, Cham, 2016.

[8] https://github.com/KaiyangZhou/pytorch-center-loss