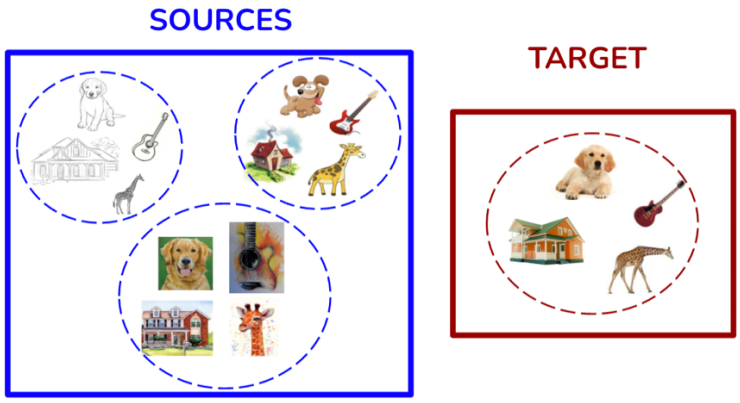
Domain-To-Text: improving Domain Generalization using Natural Language

TA: Silvia Bucci
silvia.bucci@polito.it

Problem

The ability of a model to work properly on unseen domains is crucial in many realistic scenarios: it is uncommon to know in advance the visual domain of the data with which the network has to work, and it is even more uncommon that it is provided with annotations. **Domain Generalization** (DG) is a formalization of this scenario: it is composed of several **labeled source** domains and a single **unlabeled target** domain, never seen during training.



The sources are characterized by a different domain style (e.g., sketch, cartoon, painting) and the target domain is characterized by a visual domain different with respect to all the sources (e.g., photo). Note that the distances between the target and each source domain are not equal for all the sources (e.g., the domain shift between sketch and photo is larger than the domain shift between cartoon and photo). Supposing that we trained a different network for each source domain, a solution to improve the classification performances on the target domain could be using the model trained on the source domain most similar to the target. Alternatively, we can weigh the contribution of the models trained on each source domain in function of their similarity with the target. In this project, we will investigate how a textual description of the visual domain could be helpful in measuring the source-target distance to properly weigh the contribution of each source domain in the target prediction. We suppose to have some meta-data in form of **textual description of visual domain** that we can use to compute the domain distances.

Objectives

1. Read **[1]** to become familiar with the DG setting.
2. Read **[2]** paying particular attention to the Metric Learning Approach (Section 4.2) that you will use in the next steps.
3. Run the baselines through the provided GitHub code **[3]**
4. Label a portion of the PACS **[4]** dataset following the given instruction.
5. Finetune the model provided in [2] with the new labeled dataset.
6. Repeat the evaluation done in 3.

Mandatory

Starting from the GitHub repository [3] 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 running and editing the code.

1. Evaluation to reproduce the baselines

Simply following the instruction in the README.md file you should be able to reproduce these results:

a) Simple Ensemble Baseline: it corresponds to the mean of the predictions obtained with the models trained separately on each source domain.

b) Weighted Ensemble Baseline: it corresponds to the weighted mean of the predictions obtained with the models trained separately on each source domain. The weights are obtained using the model trained in [2]. The feature vectors obtained projecting the images in the textual embedding will be used to compute the source-target distances.

	Art Painting	Cartoon	Sketch	Photo	Average
a)	67.63	57.12	60.40	94.49	69.91
b)	70.21	57.98	62.03	94.85	71.27

2. Label the PACS dataset describing the visual domain attributes

The task is to describe the appearance and the style of each image. The objects contained in the images are irrelevant to the description. You have to describe each image following these guides:

Level of details → If the image is rich in detail or it is a raw representation (e.g., high/mid/low-level)
Edges → Description of the contours of the objects (e.g., definite/precise/neat strokes, definite/precise/neat brush strokes)
Color saturation → The intensity and brilliance of colors in the image (e.g., high/mid/low, vivid colors, light reflections)
Color shades → If there are shades of colors in the image (e.g., no/yes, colorful/grayscale, etc.)
Background → Description of the background (e.g., monochrome/white/colorful etc.)
Single instance → If the image is composed by a single instance or multiple instances of the same object (e.g., yes/no, how many)
Text → If there is text in the picture (e.g., yes/no, dense/sparse text)
Texture → If there is a visual pattern that is repeated over the whole picture (e.g. yes/no, type of texture)
Perspective → If the three-dimensional proportions of the object parts are realistic (e.g. yes/no, unrealistic)

Example:

Level of details: mid-level
Edges: intense, solid yet trembling lines
Color saturation: mid
Color shades: yes, colorful
Background: cream-colored with some shades of gray
Single instance: yes
Text: no
Texture: yes, painting
Perspective: yes, realistic

N.B. You have to label at least 100 images for each domain. The labeled images have to be provided as part of the project.

3. Finetune the model provided in [2] with the new dataset

Starting from the provided model you can use the GitHub repository provided here [5] to finetune the model with the new dataset. You have to use the Metric Learning approach and save the obtained model.

4. Repeat b) of point 1 with the new finetuned weights

Once that you have the finetuned model you can simply substitute the new weights in the initial code [3] and observe the new results.

References:

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

[2] Wu, Chenyun, Mikayla Timm, and Subhransu Maji. "Describing Textures using Natural Language." In: ECCV (2020)

[3] https://github.com/silvia1993/DomainToText_AMLProject

[4] Li, Da, et al. "Deeper, broader and artier domain generalization." In: ICCV (2017)

[5] https://github.com/ChenyunWu/DescribingTextures