Domain-To-Text: Domain Generalization using Natural Language

Castrignanò Alberto s281689 | Masciavè Michele s279110 | Mercurio Lorenzo s290903 | Rainò Stefano s282436

Abstract

Domain Generalization is nowadays a crucial problem for getting effective and better predictions on real world tasks. Since the domain shift is not the same between visual domains, a measure of similarity between them can be useful for enforcing the classification performances: in this project we evaluate a such of solution, performing a metric learning approach on PACS dataset, and trying to learn a common embedding space over images and annotations such that nearby image and label pairs in that space result related to each other. Hence, the idea is weighting the contribution of each source domain according to the similarity with respect to a target one, leveraging on textual data in order to learn a reliable metric between them. These predictions are performed on one target domain by networks trained separately on the remaining three PACS visual domains. Specifically, this work presents two kind of annotations, both based on a specific template, but one leaves a free interpretation to the annotator, the other one bounds him to a predefined set of phrases to be used; with a fixed training configuration, the first approach has shown more variability and seems to present more encouraging results.

1. Introduction

Domain Generalization (DG) aims to learn a system able to perform uniformly well across multiple data distributions, extracting the most useful knowledge from samples belonging to a limited number of population sources into a single model that could generalize well on unseen target domains.

Specifically, the DG challenge is tackled starting from a fully labeled source dataset and an unlabeled set of samples from a different target domain, and developed using vary and different methods, in which source data is used for guiding the training procedure and is drawn from multiple distributions.

DG differs from **Domain Adaptation** (DA) because DA models need to be trained again when changing the target domain while the great benefit of DG should be that a new training is not required because the model may be able to generalize even on unseen domains. Note that each source is characterized by different styles (e.g., sketch, painting, photo) and the target itself belongs to a visual domain different with respect to all the sources (e.g., cartoon): this leads to different distances between the target and each source (e.g., the domain shift between photos and sketches is greater than the domain shift between photos and cartoons). Hence, the solution explored in this work is to improve the classification performances over a target domain weighting the contribution of the models trained on each source domain in function of their similarity with the target. In particular, we have investigated how **textual annotations** over a visual domain could be valuable in measuring that source-target distance. As regard the metric between visual domains, we used the metric learning approach on the training set. We then evaluated the performance of the learned metric and of the classification, and finally, we performed a weighted sum of predictions on the last visual domain, used as target, where the weights was retrieved from the learned metric.

2. Related works

2.1. Domain Generalization by solving Jig-Saw puzzles

Domain Generalization problems has been addressed in several ways. In [1] the authors

propose an end-to-end architecture which can run simultaneously:

- A **supervised object recognition** task (main objective) for understanding how to generalize across domains.
- An **unsupervised learning** task (side objective) for exploring the spatial co-location of image's patches, recovering the original image from its shuffled parts, ultimately solving a jigsaw puzzle.

Resuming the key idea exposed in [1], for which infants and toddlers learn both to categorize objects and about regularities at the same time, in this work we try to explore a new possible method to associate concepts to parts of the images, similarly to what is done in [2].

2.2. Describing Textures using Natural Language

In [2] the authors have shown that textures can provide useful cues for visual recognition tasks, analyzing the performances of several language and vision models, and finding that adopting pre-trained language networks significantly improve generalization.

Specifically, they introduced a novel dataset containing natural language texture descriptions called **Describable Textures in Detailed Dataset** (DTD^2), where each image is manually annotated with descriptions from different annotators, as a set of phrases separated by commas. They perform four different tasks:

1. **Phrase retrieval**: rank phrases that are relevant to a given image;
2. **Image retrieval by phrase**: rank the images given a phrase;
3. **Image retrieval by description**: rank the images given a complete description;
4. **Description generation**: generate a description for an input image, and then compare it against the set of its collected ones.

Authors investigate also three different techniques to learn the mapping between visual texture and natural language on DTD^2: a discriminative approach, a metric learning approach, and a generative learning approach, but they come up that the second one performs better over all the cited tasks, especially if it is combined with a BERT encoder.

For this reason we decided to rely on the metric learning as our mapping approach, and Bert as our word encoder.

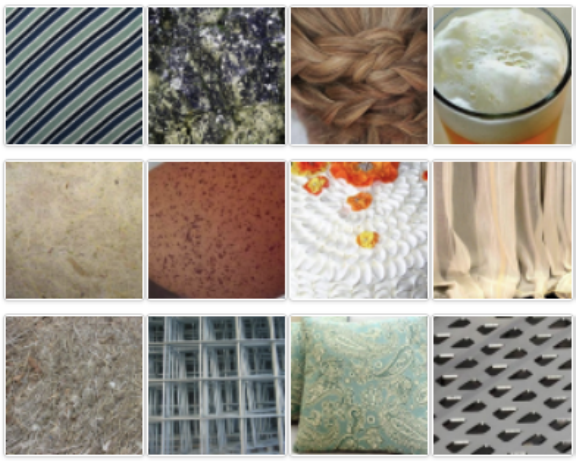

Figure 1: *The Describable Textures Dataset (DTD) is an evolving collection of textural images in the wild; this data is used for the creation of DTD^2 that describe a set of images, like the one above, with natural language descriptions*

3. Dataset configuration and annotations

3.1. Dataset

The dataset used for this project is **PACS** [3] which is a dataset developed for domain generalization tasks. It is composed by four different image domains: ArtPainting, Cartoon, Photo and Sketch. Each domain is also composed of seven different categories of objects, namely: guitar, giraffe, person, dog, house, elephant and horse. For the next annotation phase we have labeled a subset of images belonging to this dataset by balancing the number on the

various domains available. Then, we have used all the labels used for the domains both for the training phase and for the validation phase, in order to avoid having a subset of evaluation labels too small for our tasks.

3.2. Annotation Phase

First task was providing a natural language textual description to a set of images belonging to the PACS dataset [3]. Each description was made according to a template composed by nine features:

1. **Level of details**: whether the image is rich in detail or none (e.g., high/mid/low-level);
2. **Edges**: description of the contours of the objects;
3. **Color saturation**: the intensity and brilliance of colors in the image;
4. **Color shades**: if there are shades of colors (e.g., no/yes, colorful/grayscale, etc.);
5. **Background**: background description (e.g., monochrome/colorful etc.);
6. **Single instance**: if there is a single instance or multiple ones of the same object (e.g., yes/no, how many);
7. **Text**: if there is text in the picture (e.g., dense/sparse/signature/link text);
8. **Texture**: if there is a repeated visual pattern that is over the whole picture;
9. **Perspective**: if the proportions of the object are realistic (e.g. yes/no, unrealistic).

Following the template above, we have carried out two types of labeling. The first method was characterized by **manual descriptions**, leaving free interpretation over some basic guidelines agreed before starting the task. This way leads to a greater customization, hence more variability to the descriptions, but a set of words with lower occurrences. Furthermore, in this case we have labelled **400 images**.

The second method, instead, was based starting from a set of predefined phrases $\mathcal{P}$ established for each feature before starting the task. This way leads to a more restricted set of descriptions, since the annotator does not have great freedom of maneuver, except in the options available to him, creating consistent annotations, but reducing the variability over all the descriptions. A **web application** [4] was developed for achieving the task, so that the annotator could easily check the desired options between the available ones. Using this method we have labeled **515 images**, that is the entire set of images we had available, to try to understand if the number of images could be a factor for the increase in the model's performance. Figure 2 shows the user-interface of the application.

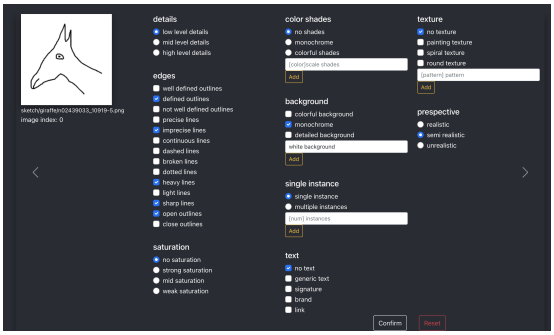

Figure 2: *Web application developed with React library, used for making annotations. The user can slide a carousel moving between the images, switch domain via a group of radio buttons, select the desired range and, of course, checking/typing the desired options for labelling the current image. Final result is directly retrieved in json format.*

3.3. Collect and analyze the labels

Once the labels for each image were collected, the annotations associated with them were stored in the form of a **list of descriptions**, with each element of the list relating to the label associated with one of the features listed above. In this way, each description in the set is considered as a **set of phrases** which are themselves ordered lists of words. Once this is done,

we have also computed some statistics on the annotations made as can be seen in Figure 3

(a)

(b)

Figure 3: *Sub-figure (a) represents the Word-Cloud generated through the descriptions associated to the manual annotation (first method) over the Art-Painting domain. Instead, sub-figure (b) shows the Word-Cloud obtained on the Art-Painting domain through the web-app annotation (second method). You can visually appreciate the different usage and distribution of words.*

4. Method

4.1. Metric Learning Approach

Second task was to study and train our model: we used an approach called **Metric Learning** to build the neural network model called **Triplet-Match** [5]. Metric learning aims to use distance metrics in such a way as to consider images with the same labels close to each other while, on the other hand, learn *not* to consider elements with different labels close to each other. In this context, it aims to learn a common embedding over images and phrases such that nearby image and phrase pairs (I, P) in that embedding space are related. The TripletMatch model used in [2] leverages on metric learning based on **triplet-loss**. In particular, given three samples drawn from the dataset, namely:

- x: a reference sample related to a particular domain (ArtPainting, Sketch, Photo or Cartoon) of the dataset
- x^+: a positive sample of the dataset, which belongs to the same domain of the reference
- x^-: a negative sample of the dataset, which belongs to a different domain with respect to the positive one and the reference

the Triplet network outputs two intermediate values: the L_2 distance between the embedded representation of the reference x and its relative positive sample x^+ and negative sample x^-, working with three instances of the same feed-forward network (with shared parameters):

$$TripletNet(x, x^+, x^-) = \begin{bmatrix} ||Net(x) - Net(x^-)||_2 \\ ||Net(x) - Net(x^+)||_2 \end{bmatrix}$$

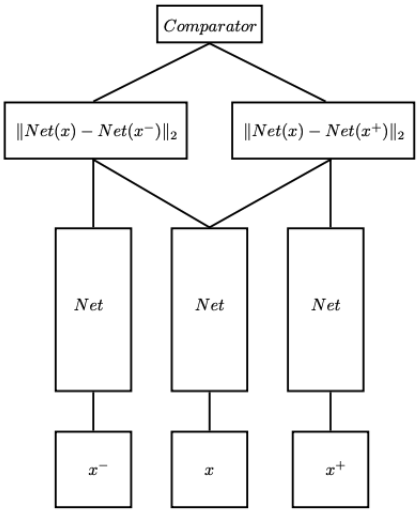

Figure 4: *Triplet network structure. The model encodes the pair of distances between each of x^+ and x^- against the reference x.*

This model is used starting from an annotation (I, P) consisting of a positive (image, phrase) pair, then it samples from the training

set a negative image I' for P, and a negative phrase P' for I, in order to calculate **two losses**:

- The first one from **negative phrase**:

$$L_p(I, P, P') = max(0, 1 + ||\psi(I) - \phi(P)||_2^2 - ||\psi(I) - \phi(P')||_2^2)$$

- The second one from **negative image**:

$$L_p(P, I, I') = max(0, 1 + ||\psi(I) - \phi(P)||_2^2 - ||\psi(I') - \phi(P)||_2^2)$$

In both losses the objective is to learn embeddings ψ and ϕ that minimize the **mean sum of the losses** on the training set, where $\psi(I)$ in the formulas is the embedding of an image and $\phi(P)$ is an embedding of a phrase in $\mathbb{R}^d$.

For embedding images, the authors extract features from layer 2 and 4 from ResNet101 and they also add an additional linear layer with 256 units. At the end, they consider to use different types of encoders, which are: Mean-pooling, LSTM, ELMo, BERT. for our tasks we use **BERT** (Bidirectional Encoder Representations from Transformers), which uses its own tokenizer, and outputs the average of last hidden states of all tokens in the phrase P.

4.2. BERT encoder and ResNet encoder

Bidirectional Encoder Representations for Transformers is a Neural Network, recently introduced in [6], designed for pre-training deep bidirectional representations from unlabeled text by jointly conditioning on left and right context in all layers. As a result of the training process, BERT learns contextual embeddings for words. Wu et al. [2] use a pre-trained BERT model with a $H = 768$ shaped vector for each input token (the hidden size). Through the **BertTokenizer** the encoder considers a **"sentence"** as an arbitrary division of contiguous text, while a **"sequence"** as an input token sequence for BERT, which can be considered either as a single sentence or as a pair of two sentences put together.
Subsequently to pre-process the input, the first token of each sequence is considered to be a classification token **[CLS]**, while each sentence is separated from the next with a special token **[SEP]**. Then, the words are converted into **embeddings**, in such a way that the model can understand the semantic importance of a word in the numeric form. Three embeddings are eapplied:

- **Token Embeddings**: which represents a tokenization strategy applied for a good balance of vocabulary size and out-of-vocab words;
- **Segment Embeddings**: which represents the sentence number that is encoded into a vector. At this stage, the model needs to know whether a token belongs to sentence A or sentence B in BERT;
- **Position Embeddings**: which is used to model how a token at one position attends to another token at a different position.

In conclusion, all the three embeddings mentioned above are summed element-wise (as represented in figure 5) to produce a single representation with the shape of H, and then the output of the hidden state is used in order to compute their distance from the encoding of the images. This last encoding is done through a **ResNet encoder** based on a ResNet backbone from where the average of the layers output is extracted.

4.3. Baseline and Metrics

Third task was to evaluate the models trained on the different sources domain with one target domain each time. After that, we collected the accuracy performances which we wanted to improve. To do this we loaded the various models trained separately for one of the four domains and evaluated these models based on the remaining domain (e.g. source domains Cartoon, ArtPainting, Sketch and target one Photo). The data collected after the training is related to two metrics:

- **Accuracy mean**: which represents the mean of the predictions obtained with the models trained separately on each source domain;

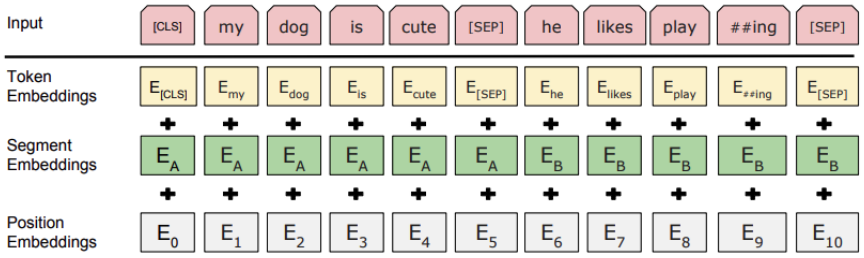

Figure 5: *BERT embeddings representation.*

- **Accuracy text domain embedding**: which represents the weighted mean of the predictions obtained with the models trained on each source domain;

The first metric is computed by the average regarding the evaluation of the pre-trained models.

The second metric, instead, requires the computation of the distances between each source and the target domain, in the form of weights. To extract these weights through the Triplet-Match model, it was necessary to encode the images in an embedding space and then calculate the similarity between the image considered, related to the target domain, and the average in each of the remaining source domains. Then the final step was to compute a weighted average, considering the calculated similarities as weights, between the output of the model and the similarities.

Starting from these results, in the following steps, we have **fine-tuned** the model through the use of weights related to the annotations made on the PACS dataset. The aim was to obtain weights that **best describe the distances between the various domains**, and the objective was to increase the accuracy of the predictions obtained by running the evaluation after fine-tuning.

5. Training

The training was performed in order to collect the new weights used for fine-tuning the model. The approaches cited on table 1 rely on that same preliminary training configura-

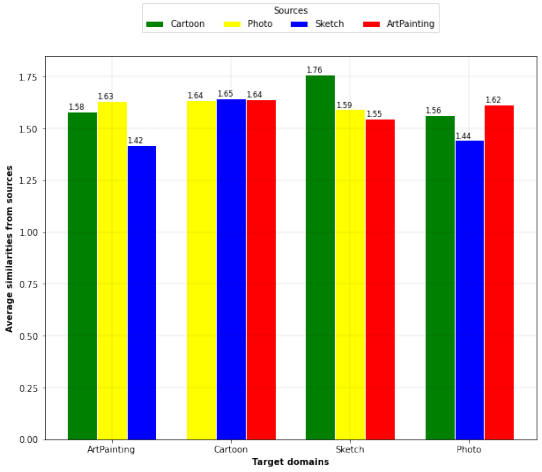

Figure 6: *Similarity weights comparison between domains, done after the evaluation made by the baseline.*

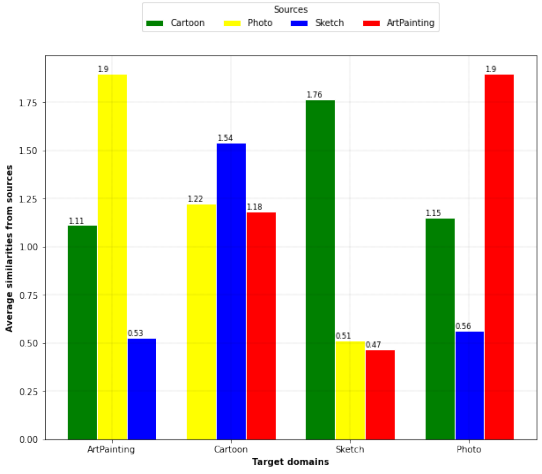

Figure 7: *Similarity weights comparison between domains, done after the fine-tuning made by the best configuration of WEMA (see table 1).*

Method	ArtPainting	Cartoon	Sketch	Photo	Average
Simple Ensamble Baseline	67.63	57.12	60.40	94.49	69.91
Weighted Ensamble Baseline	70.21	57.98	62.03	94.85	71.27
WEWA	70.46	**60.79**	63.35	97.07	72.92
WEMA	**72.08**	60.58	**63.65**	**97.31**	**73.41**

Table 1: *Comparison between the best configurations for manual annotation and web application annotation, tested on each target domain. Simple Ensamble Baseline is the baseline related to the accuracy mean metric. Weighted Ensamble Baseline is the baseline related to the accuracy text domain embedding metric.* ***WEWA*** *and* ***WEMA*** *stand for "Weighted Ensamble Web Annotation" and "Weighted Ensamble Manual Annotation" which are related to the same metric of the Weighted Ensamble Baseline, but the one over manual descriptions, and the other one over web application descriptions, on PACS dataset. (see 4.3)*

Method + Distance metric	ArtPainting	Cartoon	Sketch	Photo	Average
WEMA + *Euclidian Squared Distance*	70.02	57.72	62.05	94.79	71.15
WEMA + *Euclidian Distance*	69.97	59.22	62.61	95.09	71.72
WEMA + ***Cosine Similarity***	**72.08**	**60.58**	**63.65**	**97.31**	**73.41**

Table 2: *Comparison table of the distance applied on the embeddings for the WEMA method (see 4.3)*

tion, using the following hyper-parameters: $batchsize = 16$, maximum $epochs = 6$, learning-rate $lr = 0.00001$ with a decay gamma $g = 0.1$ and weight decay $d = 1e - 6$, Adam as optimizer. The TripletMatch model is characterized by BERT as Word-Encoder, with phrases as input, and cosine-similarity as distance metric. See our GitHub repository [7] for more information.

6. Experiments

In table 1 we have collected the results obtained from our experiments carried out by training the TripletMatch model on the PACS dataset with the best configuration of hyper-parameters, and re-evaluating the baseline by collecting the weights obtained from the training. In particular, all the configurations tested included an initialization with the weights obtained on the training carried out on the DTD^2 and with the subsequent fine-tuning on our dataset. We tried this approach in relation to both label techniques mentioned in section 3.2. In table 2 instead, we tried **different distance metrics** for the manual annotation, which is the one that has given us the best results. In this case we have computed three different types of embeddings distances for the computation of the losses, in order to verify which one is better; they are: **Euclidian Distance**, **Euclidian Squared Distance** and **Cosine Similarity**.

The figure 6 represents the similarities between domains according to the baseline evaluation, instead, figure 7 shows the graph relating to the similarities between domains obtained after the fine-tuning.

7. Results

As shown from the tables, a slight improvement has obtained compared to the baseline, both for WEMA and WEWA. The results obtained with WEMA are the best, and this could be due to two factors:

- The greater number of annotations made through web applications that could have led to a consequent overfitting in relation to the model
- The presence of greater variability in relation to manual annotations which allowed

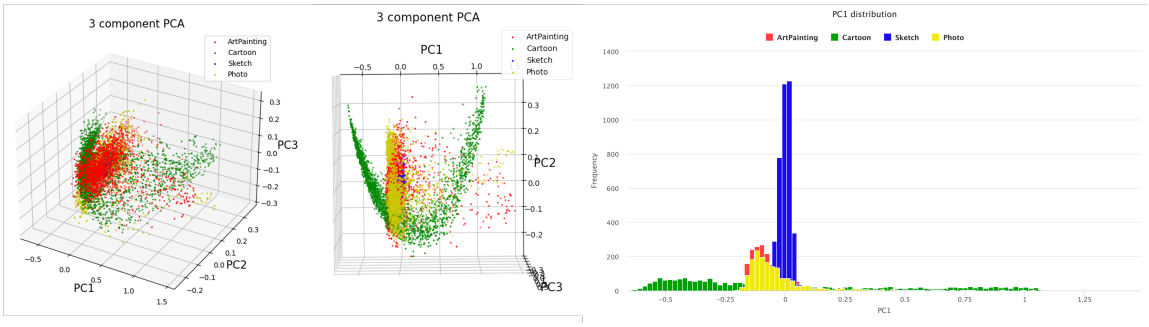

Figure 8: *PCA(m=3) static view of PACS textual embedding space and PC1 (principal component) values distribution for each class (bin=100). It is possible to appreciate how some domains result close to each other, confirming what is described by the weights in figure 7.*

the model to discriminate the available data more correctly.

The figure 7, instead, shows how the similarity weights are consistent with the annotations provided, showing how the results obtained improve the similarities between domains already present in the baseline. For instance, it is reasonable to think that the Sketch target domain is the furthest from the ArtPainting source domain and the closest to the Photo source domain. This is also visible in figure 8, both on the PCA points distributions (red points and yellow ones), and on the first principal component (PC1) data distributions (red histogram partially over the yellow one). Particularly interesting is the cluster in figure 9, representing a zoom on Sketch target domain on the PCA textual embedding: points distribution is basically consistent with what shown in figure 7 and suggests a particularly discriminating annotation for this target. A possible further improvement could be achieved by increasing the number of manual annotations or by changing the way these annotations are taken (see 3.2).

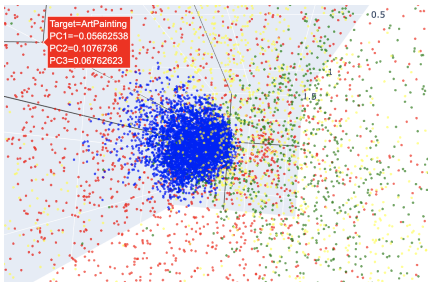

Figure 9: *PCA(m=3) zoom on Sketch target domain textual embedding.*

References

[1] Tommasi et al. "domain Generalization by solving JigSaw puzzles." CVPR 2019

[2] Wu et al. “Describing Textures using Natural Language.” ECCV 2020

[3] Da Li et al. "Deeper, Broader and Artier Domain Generalization." ICCV 2017

[4] https://github.com/Michele-Masciave/aml-describing-images

[5] Hoffer et al. "Deep Metric Learning using Triplet Network." ICLR 2015

[6] Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." Google AI Language, 2019

[7] https://github.com/steo13/aml-domain2text-project