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

Pietro Borgaro Lorenzo Cravero Roberto Torta

Politecnico di Torino, Italy

{s292501, s290265, s290184}@studenti.polito.it

https://github.com/git-pushz/DomainToText_AMLProject

Abstract

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). Since 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) we can think of two ways to improve classification performances on the target domain supposing that we trained a different network for each source domain. First solution could be to use only the model trained on the most similar domain with respect to the target one. Another solution, which will be chosen in this project, is to 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 created some meta-data in form of textual description of visual domain that we can use to compute the domain distances.

1. Introduction

In this project we tried to improve the performance in object recognition in a DG setting by using a textual description of the visual domain. We added annotations to 100 images for each domain of the PACS dataset (more on this in paragraph 3) to see if these descriptions of the images could be helpful in assigning more significant weights to each source domain to better reflect its distance from the target.

	Simple	Weighted
Art Painting	67.63	70.21
Cartoon	57.12	57.98
Sketch	60.40	62.03
Photo	94.49	94.85
Average	69.91	71.27

Table 1. Benchmarks that we try to improve

Our goal was to overcome the baseline results (Table 1) obtained computing the similarities weights with the TripletMatch model introduced in "Describing Textures using Natural Language" [2] trained on the original dataset DTD2 (more on this in paragraph 2.2). To reach our goal we fine-tuned the TripletMatch model, which is based on the Metric Learning Approach, with our aforementioned dataset composed of annotated images from four different visual domains. With this new model we then repeated the classification task managing to get slightly better performances in most of the domains.

2. Related Works

2.1. Domain Generalization by Solving Jigsaw Puzzles

This project was inspired by the paper "Domain Generalization by Solving Jigsaw Puzzles" [1], which aims at solving the Domain Generalization problem. The idea of Domain Generalization is to learn from one or multiple source

training domains, to extract a domain-agnostic model which can be applied to an unseen target domain. The main challenge is to extract the most useful and transferable general knowledge from samples belonging to a limited number of population sources.
Domain Generalization differs from Domain Adaptation because in DA some samples from the target domain are available at training time (even if unlabeled) and are used to guide the source training procedure. However DA models have to be trained again when changing the target domain while the great benefit of DG is that this new training is not required because the model should be able to generalize even on unseen domains.
This paper propose a model that tries to tackle the DG problem with a multi-task process. The model is trained to solve two task simultaneously: object classification and patch reordering (jigsaw puzzle). Jigsaw puzzles is an unsupervised task which consist of recovering an original image from its shuffled parts thus learning about spatial co-location of image parts. This task is optimized jointly with the object classification one, with the goal of improving generalization, in fact the two tasks shares the same network backbone in the proposed architecture.

2.2. Describing Textures using Natural Language

To improve the results of the domain generalization problem we started from the paper "Describing Textures using Natural Language" [2]. This paper present a novel dataset of textures with natural language descriptions and analyze the performance of several language and vision models. The new dataset is called DTD2 and contains descriptions of each image in DTD dataset. The descriptions are not a grammatically coherent sentence but they are a set of phrases like this one: image1 = [circular overlapping red yellow green twisted, spiral, round, patches, rings, multicolored]. The paper then considers different tasks and evaluation metrics. The tasks are: phrase retrieval (given an image rank the phrases that are relevant to the image), image retrieval from a phrase (given a phrase rank the images), image retrieval from a description (given a description rank the images) and finally description generation (generate a description for an image and compare it with the ones from the dataset). Authors investigate three techniques to learn the mapping between visual texture and natural languages on the dataset: a discriminative approach (that treat each phrase as a binary attribute and train a multi-label classifier to map the images to phrase labels), a metric learning approach (that aims to learn a common embedding over the images and phrases such that nearby image and phrase pairs in the embedding space are related) and a generative learning approach (that combines a convolutional network to encode input images with an attention-based LSTM decoder to generate descriptions). They arrived at the conclusion that the metric learning approach provides better results in all the tasks, especially if it is combined with a BERT encoder. For this reason this approach will be the one used in our project.

3. Dataset and Annotations

3.1. Dataset

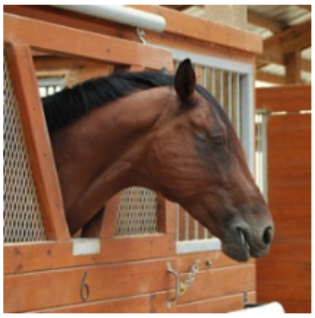
For this project we used the PACS dataset [4]. PACS is an image dataset for domain generalization. It consists of four domains, namely Photo (1,670 images), Art Painting (2,048 images), Cartoon (2,344 images) and Sketch (3,929 images).
Each domain contains seven categories for the object detection tasks (giraffe, guitar, person, dog, house, elephant, horse). Each domain contains images with different definition, shapes, colors, background, texture, number of instances to increase the variability for each object.

3.2. Annotations

We proceeded to label 400 images (100 for each domain), randomly chosen from the PACS dataset, trying to describe in a textual way the appearance and style of each image (examples in figure 1). We focused our attention on 9 fields:

- level of detail → 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...)
- number of 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)

(a) Level of details: low-level; Edges: precise lines; Color saturation: mid, light reflections; Color shades: yes, colorful; Background: white; Single instance: yes; Text: no, Texture: without texture; Perspective: no, unrealistic.



(b) Level of details: high-level; Edges: intense, solid lines; Color saturation: high, vivid color; Color shades: yes, colorful; Background: yes, colorful; Single instance: yes; Text: no; Texture: without texture; Perspective: yes, realistic.

(c) Level of details: mid-level; Edges: trembling brush strokes; Color saturation: mid colors; Color shades: yes, colorful; Background: colorful; Single instance: no, 3; Text: no; Texture: brush strokes; Perspective: yes, realistic.

(d) Level of details: low-level; Edges: light, irregular lines; Color saturation: no colors; Color shades: no; Background: without background; Single instance: yes; Text: no; Texture: without texture; Perspective: no, unrealistic.

Figure 1. One example for each domain

4. Method

4.1. Baseline

Our first step in the project was to obtain the benchmarks that we would have then tried to improve. We evaluated the models trained on the different sources domain with one target domain each time. For instance when we choose Art-Painting as target we load the models trained separately on Cartoon, Photo and Sketches and we evaluate those models with the samples taken from the ArtPainting domain. We then compute two results:
a) "Simple Ensemble Baseline" which corresponds to the mean of the predictions obtained with the models trained separately on each source domain.
b) "Weighted Ensemble Baseline" which corresponds to the weighted mean of the predictions obtained with the models trained separately on each source domain.

While a) is a simple mean that requires nothing more than evaluating the pre-trained models and calculating the average, b) requires some weights to be established to represent the distances between each source domain and the current target. Firstly, to obtain those weights, we used the model introduced in [2] named "TripletMatch". By using this pre-trained model we were able to encode the images (both from the sources and the target) in an embedding space. From that for each image in the target domain we computed the similarity between the image and the mean in each source domain in order to determine a weight to assign to each domain. With these three weights for each target image we computed the b) value.

The main idea of this project was to obtain those weights from an unsupervised training on some natural language textual description of a part of the images of the PACS dataset. To do so we created a dataset with annotations for 400 images taken from PACS as described in paragraph 3 and we fine-tuned the model of [2] with this little dataset of our own. We were aiming to obtain more significant weights that could better describe the distances between the domains. With the new fine-tuned model we ran again the baseline and we obtained the new results that we will show in paragraph 5.

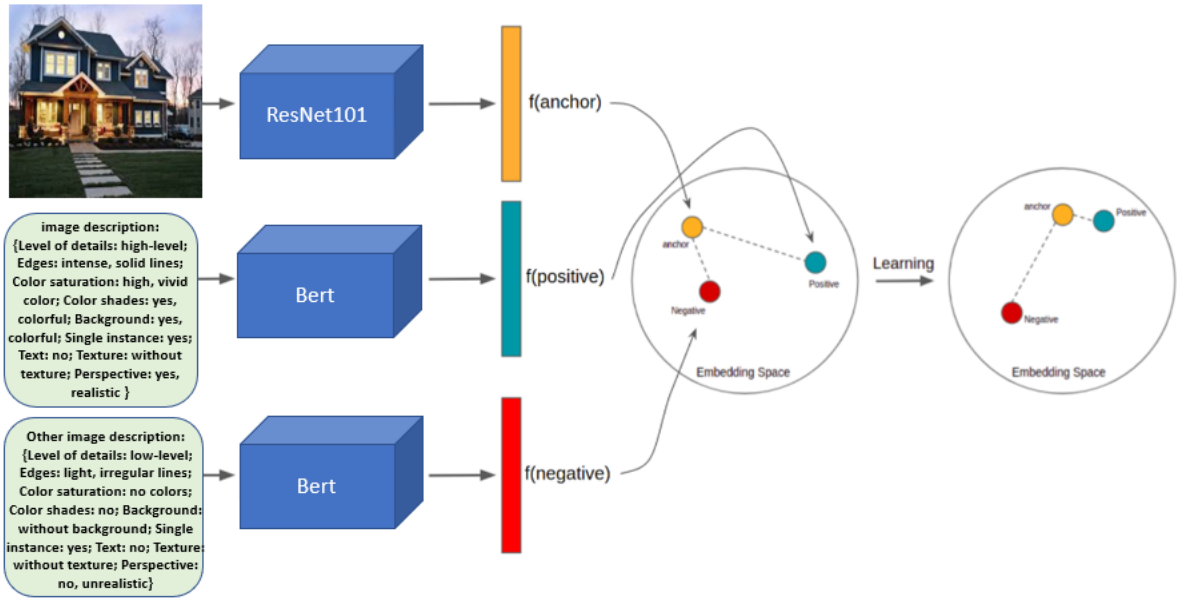


Figure 2. Metric learning approach used in this project

4.2. Metric Learning Approach

Metric learning is a method of determining similarity or dissimilarity between items based on a distance metric. Metric learning seeks to increase the distance between dissimilar things while reducing the distance between similar objects in the embedding space. In the context of "Describing Textures using Natural Language" [2] the metric learning approach aims to learn a common embedding over the images and phrases such that nearby image and phrase pairs in the embedding space are related (Figure 2). The authors of Describing Textures use the metric learning approach based on triplet-loss which is implemented in the Triplet Network [6].

To explain Triplet Network we start from considering three samples taken from the dataset (triplet):
x is our first sample from the dataset also known as anchor or reference,
x+ is a second sample from the dataset and it is related to the first one (e.g. same class) and is known as positive,
x- is a third sample from the dataset and it's not related to the first one (e.g. different class) and is known as negative

A Triplet network is comprised of 3 instances of the same feed-forward network (with shared parameters). When fed with 3 samples, the network outputs 2 intermediate values - the L2 distances between the embedded representation of two of its inputs from the representation of the third.

This model is used in Describing Textures by calculating two losses:

$$TripletNet(x, x^-, x^+) = \begin{bmatrix} \|Net(x) - Net(x^-)\|_2 \\ \|Net(x) - Net(x^+)\|_2 \end{bmatrix} \in \mathbb{R}^2_+ .$$

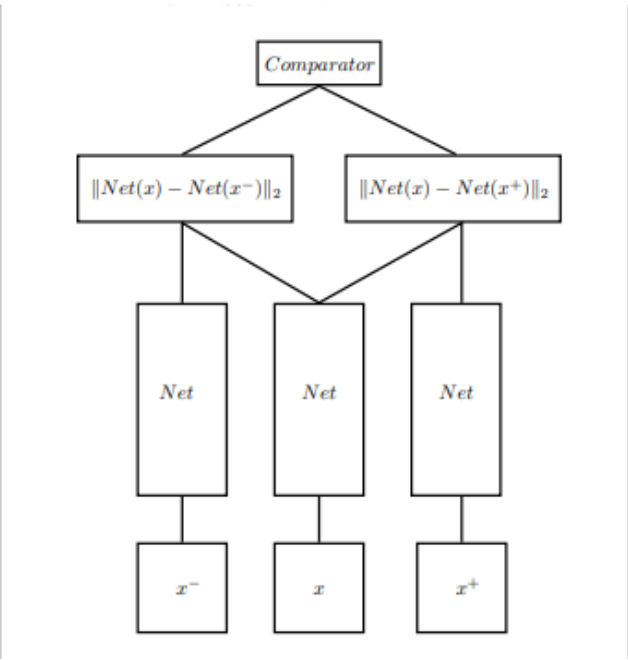


Figure 3. In words, this encodes the pair of distances between each of x+ and x- against the reference x.

1. negative phrase loss: x is an image (I), x+ is the phrase or description associated with the image (P), x- is a phrase

or description not associated with the image (P')

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

2. negative image loss: x is a phrase or description (P), x+ is the image described by the phrase (I), x- is an image which is not described by the phrase (I')

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

The objective is to minimize both losses.

For embedding images, they use activations from layer 2 and layer 4 of ResNet101 with mean-pooling over spatial locations. (Features from layer 2 and 4 from ResNet101).

For embedding language (phrase or description) they use BERT encoder. Bidirectional Encoder Representations from Transformers (BERT) is a transformer-based machine learning technique for natural language processing (NLP) pre-training developed by Google [7].

4.3. BERT

BERT architecture is essentially made up of stacked up encoder layers. It provides both contextual and bidirectional embedding, the first meaning that the embedding of a word depends on the context of words around it and the latter meaning that the sentence is not read by the model sequentially but all at once.
The paper use a pre-trained BERT, which provides, for each input token(word) in a sequence, a H = 768 shaped vector. The pre-training is composed of two unsupervised tasks. For the first one random words of the input sequence are masked and the goal is to predict them (the mask can either hide the word, substitute it with a random one or keep it untouched). This first task is useful for the model to learn words' context, the second one aims at understanding the relationship between two sentences. It is called Next Sentence Prediction(NSP): during training two sentences A and B are chosen such that 50% of the time B is the actual next sentence that follows A and 50% of the time it is a random sentence from the corpus. The goal is to correctly classify the two sentences as related or unrelated.
To help the model distinguish between the two sentences in training, the input is processed in the following way before entering the model:
1) A [CLS] token is inserted at the beginning of the first sentence and a [SEP] token is inserted at the end of each sentence.
2) A sentence embedding indicating Sentence A or Sentence B is added to each token. Sentence embeddings are similar in concept to token embeddings with a vocabulary of 2.
3) A positional embedding is added to each token to indicate its position in the sequence. The concept and implementation of positional embedding are presented in the Transformer paper[8].

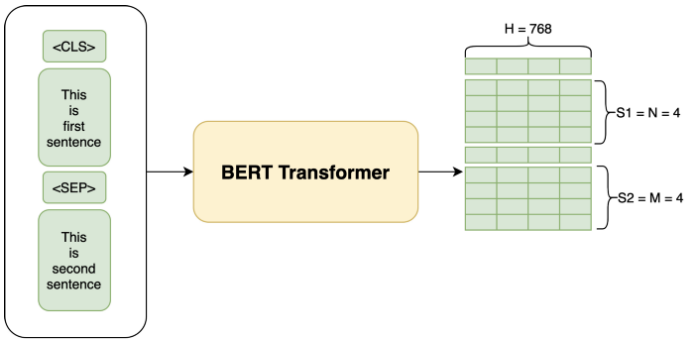


Figure 4. Bert tokenize and then encode words sequences

5. Experiments

In Table 2 we show the results obtained after two different training over the model TripletMatch with our dataset of annotated PACS images.
In column Domain-To-Text (A) the model is initialized with the weights obtained over the training on the dataset DTD2 and then fine-tuned on our dataset. In column Domain-To-Text (B) we didn't pre-load any weights and we directly trained the model on our dataset. In Table 3 we show the average of the similarity weights obtained during the evaluation on each target domain. On each cell the value represents the similarity between the target (the row) and the source (the column) domain.

6. Conclusions

We obtained a slight improvement with respect to the baseline benchmark only in the Domain-To-Text(B) experiment, where we fully trained the TripletMatch model. Given the low number of samples (100 annotated images for each domain) this could be caused by the model memorizing and consequently overfitting over the PACS image dataset, so the better accuracy doesn't necessarily translate to a better model. A test on images from a different dataset could solidify the results. As shown in table 3, the similarity weights obtained are what one would expect for these specific domains. It is reasonable that when the target domain is Cartoon, the closest source is Sketch, at the same time ArtPainting results closer to Photo. This kind of similarity is already present in the ensemble baseline, however, our model was able to make closest domains even closer. Our results could have been improved if we trained the model with a bigger dataset of annotations, and also increasing the number of descriptions for each image could have helped. Another approach to try to get better performances could have been to force the Triplet Network model to choose

	Simple ensemble baseline	Weighted ensemble baseline	Domain-To-Text (A)	**Domain-To-Text (B)**
Art Painting	67.63	70.21	69.24	**70.26**
Cartoon	57.12	57.98	50.68	**58.96**
Sketch	60.40	62.03	62.05	**63.58**
Photo	94.49	94.85	94.79	**95.39**
Average	69.91	71.27	70.89	**72.05**

Table 2. Results comparison

Method	Target	Art Painting	Cartoon	Sketch	Photo
Weighted Ensemble Baseline	Art Painting	-	1.58	1.42	1.63
	Cartoon	1.64	-	1.65	1.63
	Sketch	1.55	1.76	-	1.59
	Photo	1.62	1.56	1.44	-
Domain-To-Text (B)	Art Painting	-	1.43	1.27	1.57
	Cartoon	1.47	-	1.7	1.39
	Sketch	1.34	1.85	-	1.41
	Photo	1.58	1.38	1.32	-

Table 3. Similarity weights comparison

a sample from another domain as negative sample, maybe even within the same class. This consideration arises from the fact that in the DTD2 dataset there is no notion of class or domain, so the negative sample is chosen randomly (just checking that is different with respect to the positive one).

7. References

0 https://github.com/git-pushz/DomainToText_AMLProject

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

6 Hoffer, E., Ailon, N.: "Deep Metric Learning using Triplet Network". In: International Workshop on Similarity-Based Pattern Recognition. pp. 84–92. Springer (2015)

7 Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: "BERT: Pre-training of deep bidi-rectional transformers for language understanding". (2018)

8 Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. "Attention is all you need". In Advances in Neural Information Processing Systems, pages 6000–6010