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

Advanced Machine Learning

Politecnico di Torino - A.A. 2021/2022 Winter Session

Leonardo Iurada s291018

Stefano Strippoli s282870

Andrea Di Domenico s287639

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. In this project we evaluate the possibility of using textual description of visual domains of images to perform metric learning on the PACS dataset. Specifically, since the domain shift is not the same betweeen visual domains, the aim is to learn a reliable metric between them to weight predictions performed on a target visual domain of the PACS dataset. These predictions are performed by models trained separately on the other 3 visual domains. Encouraging results are presented on using textual descriptions of visual domains in natural language to enhance Domain Generalization via metric learning.

1. Introduction

Domain generalization (DG) aims to incorporate knowledge from multiple source domains into a single model that could generalize well on unseen target domains. 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). 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 managed to manually write textual descriptions in natural language for 400 images belonging to the PACS dataset [1] using the following template:

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

These 400 images (100 from each visual domain) and relative textual descriptions represent our dataset to work with. We then performed a 60-20-20 split in order to build a train, validation and test set respectively, making sure the proportionality among object categories and visual domains remained the same on the different splits.

In order to learn a reliable metric between visual domains we used the metric learning approach [3] on our training split. We then evaluate the performance of the learned metric (on the test split) and of the classification (training separately 3 classifiers on 3 different visual domains, used as sources). Then we perform a weighted sum of predictions on the last visual domain, used as target. Weights are given by the learned metric.

2. Related Work

Metric Learning. 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. Based on the work of Wu et al. [3] metric learning can be based on the standard triplet-loss for learning a joint embedding of images and phrases such that one could retrieve the nearest images from a specific phrase or, viceversa, retrieve the nearest phrases from a specific image. In the cited work, Wu et al. used this approach to generate textual descriptions in natural language for image textures. In our project, instead, we will use textual descriptions' embeddings as a sort of anchor to space reasonably the image embeddings across the visual domains of the PACS dataset.

Domain Generalization. DG is a zero-shot problem as performance is immediately evaluated on the target domain with no further learning. Despite different methodological tools (SVM, subspace learning, autoencoders, etc), existing methods approach DG based on a few different intuitions. One is the idea of generating a domain agnostic classifier, for example by asserting that each training domain's classifier is a domain-specific weight vector. The resulting domain-agnostic weight vector can then be extracted via a weighted sum of domain-specific weight vectors and applied to held out domains. Our approach lies in this latter category. Several works have reduced the problem to the domain adaptation (DA) setting where a fully labeled source dataset and an unlabeled set of examples from a different target domain are available. This is the setting where we will further investigate, by also enriching information via textual descriptions in natural language of visual domains.

3. The Metric Learning Approach

Based on section 4.2 of [3], the metric learning approach aims at learning a joint embedding over the images and phrases such that nearby images and phrase pairs in the embedding space are related. We aim at learning a reliable measure of distance based on visual features of the images. Specifically, we work with image and phrase pairs where each image belongs to a visual domain and the relative phrase is the textual description of the visual features of the image (so, it describes its visual domain). We adopt the standard metric learning approach based on triplet-loss [3].

Consider an embedding of an image $\psi(I)$ and of a phrase $\phi(P)$ in $\mathbb{R}^d$. Denote $\|\psi(I) - \phi(P)\|_2^2$ as the squared Euclidean distance between the two embeddings. Given an annotation (I, P) consisting of a positive (image, phrase) pair, we sample from the training set a negative image I' for P, and a negative phrase P' for I. We consider two losses; one from the negative phrase:

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

and another from the negative image:

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

The metric learning objective is to learn embeddings ψ and ϕ that minimize the loss $L = \mathbb{E}_{(I,P),(I',P')}(L_p + L_i)$ over the training set.

For embedding images, we use the encoder part from ResNet101, specifically, we take activations from layer 2 and layer 4 of ResNet101. We add an additional linear layer with 256 units resulting in the embedding dimension $\psi(I) \in \mathbb{R}^{256}$. For embedding phrases (textual descriptions) we use pre-trained BERT model with its own tokenizer, and outputs the average of last hidden states of all tokens in the phrase P. To compute the final embedding of the phrase $\phi(P)$, we add a linear layer to map the embeddings to 256 dimensions compatible with the image embeddings. The pre-trained BERT model is frozen during training.

3.1. Training Details

Initial setup. We took the initial training setup as implemented in [2] to implement the training strategy we followed. We added data augmentation techinques on images (specifically, we used random resized crop with random horizontal flipping) during training.

Online hard-negative mining. We started from the provided *'outputs/triplet_match/BEST_checkpoint.pth'* and in the first part of the training we finetuned these weights using online hard-negative mining sampling on the training split. This technique aims at selecting P' such that $\phi(P')$ is close to $\psi(I)$ and I' such that $\phi(P)$ is close to $\psi(I')$, given (I, P) positive pair. At each iteration, we perform this kind of mining to build our minibatch. Mathematically: $\forall (I, P), P' = \arg\min_{P^*} \|\psi(I) - \phi(P^*)\|_2^2$ and $I' = \arg\min_{I^*} \|\psi(I^*) - \phi(P)\|_2^2$
This makes the problem harder but it allows for having more "interesting" tuples to train the network, since if we just sampled randomly we may get I', P' already well spaced from P, I respectively. The latter case may

lead to non-meaningful embeddings (the network is more prone to learn a mapping s.t. $\|\psi(I) - \phi(P)\|_2^2 = 0$, $\|\psi(I') - \phi(P)\|_2^2 = 0$ and $\|\psi(I) - \phi(P')\|_2^2 = 0$, which means that embeddings are mapped on a single point). Or another occurrence may be that the training doesn't converge, due to too much variability in the generated mini-batches. So online hard-negative mining allows for learning meaningful embeddings since we cycle through balanced mini-batches of samples from the different visual domains, and we always take the most "interesting" tuple to train with. We trained with Adam optimizer (LR=0.000002), stopping when the loss consolidated and the mAP on the validation split stopped improving.

Online batch-hard mining. We started from the *'BEST_checkpoint.pth'* of the previous step (Online hard-negative mining) and we finally finetuned these weights using online batch-hard mining sampling on the training split. This technique not only aims at selecting P' such that $\phi(P')$ is close to $\psi(I)$ and I' such that $\phi(P)$ is close to $\psi(I')$, given (I, P) positive pair, but also selects only $\psi(I)$ and $\phi(P)$ that are far apart. At each iteration, we perform this kind of mining to build our minibatch. Mathematically:
$(I, P) = \arg\max_{(I^*, P^*)} \|\psi(I^*) - \phi(P^*)\|_2^2$,
$P' = \arg\min_{P^*} \|\psi(I) - \phi(P^*)\|_2^2$
and
$I' = \arg\min_{I^*} \|\psi(I^*) - \phi(P)\|_2^2$

This makes the problem even harder but allows for finalizing the model with what was poorly learned during the previous step. We trained with Adam optimizer (LR=0.00001), stopping when the loss consolidated (much below 1.0) and the mAP on the validation split stopped improving.

4. Results

4.1. Evaluation of the Learned Metric

mAP. To evaluate the learned metric we started from mAP_{i2p} of phrase retrieval (given an image) and mAP_{p2i} of image retrieval (given a phrase). Specifically, given an image, the goal is to rank phrases $p \in \mathcal{P}$ that are relevant to the image, since there are 100 for each visual domain. Here $\mathcal{P}$ is the set of all possible phrases. We take as ground truth just the corresponding phrase to make the evaluation process more strict, since the textual embeddings with almost no training were already reasonably spaced, meaning that the labeling process was meaningful.

Given a query phrase, mAP_{p2i} evaluates the task of retrieving images. When taking phrases as the query, we consider all phrases $p \in \mathcal{P}$ as before and ask the model to rank all images in the test or validation set. Again, we take as ground truth just the corresponding image.
Finally, to retrieve the overall improvement we simply sum

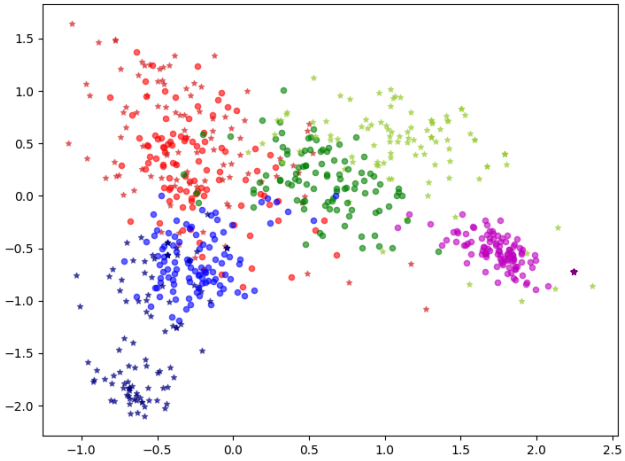

Figure 1. PCA(m=2) of image and textual embeddings. The stars (⋆) refer to textual embeddings, while the dots (•) refer to image embeddings. Visual domains: Art Painting, Photo, Cartoon, Sketch

mAP_{i2p} and mAP_{p2i}. We generically call the sum of these 2 metrics: mAP, which ranges between 0.0 and 1.0 (of course, the higher, the better).

Validation scores. On the validation split, during the training of our model, specifically during the hard-negative mining part, we achieved an overall mAP of 0.436. Continuing the training using that methodology lead to lower mAPs. Then, during the batch-hard mining part, we achieved an overall mAP of 0.549. Again, continuing the training using that methodology lead to lower mAPs.

Test scores. Passing to the evaluation part (on the test split), we achieved a mAP of 0.613, which is significantly better than the score on the validation set. This could be due to the fact on how the splits were made in the beginning.

Visualizing clusters. To conclude this section, we also tried to visualize the goodness of the learned metric. We took all 400 pairs, computed both image and textual embeddings, applied PCA just on textual embeddings to compute the most discriminative directions and projected both image and textual embeddings on these 2 most discriminative directions. In Figure 1. we can observe the learned embeddings. Specifically, we see 2 main things:
1) that visual domains' distances are reasonable. Photos are more similar to art paintings, cartoons are between art paintings and sketches, and finally sketches and photos are far apart.
2) For cartoon, sketch and photo visual domains the image and textual embeddings only partially overlap. This could

	Art Painting	Cartoon	Sketch	Photo	AVG
(a)	67.63	57.12	60.40	94.49	69.91
(b)	70.21	57.98	62.03	94.85	71.27
(ours)	**72.12**	**58.61**	**63.25**	**95.27**	**72.31**

Table 1. Results on the Test set. Using (a) we just take the average of the predictions made by the 3 classifiers on the target domain. (b) weights the predictions using the provived BEST_checkpoint.pth. (ours) refers to our model finetuned with the metric learning approach.

mean that doing more training could have lead to better results, but we cannot say it for sure since we are just evaluating a projection onto a 2D surface. As a matter of fact, we are projecting on the 2 most discriminative directions of the textual embeddings so we are not really considering how image embeddings get transformed by the projection. However, it is clear that enriching data with textual descriptions improves the learning process, since if we consider textual and image embeddings separately (even if at slightly different scales) they form similar clusters that are also spaced in the same way.

4.2. Evaluation of the Classification

Having seen the learned metric results, we now focus on evaluating how much the classification improves using the metric learning approach.

Classification is made via models trained on 3 source visual domains of the PACS dataset and then applied on the target domain (the last one). Specifically, these classifiers consist in a linear layer (that outputs predictions for the 7 object classes of the PACS dataset) on top of a pretrained ResNet18 (used as feature extractor). Finally the final outputs get passed inside a softmax function.

Method (a). Consists in taking the arithmetical mean of the 3 predictions made on the target domain, without using anything else.
Method (b). Consists in taking the weighted average of the 3 predictions made on the target domain. To compute the weights, for each image of the source domains, we calculate the image embedding (we aggregate by summing in order to have 3 vectors representative of the 3 source visual domains) and then we use cosine similarity between the image embedding vector of the target and the vectors of the 3 source visual domains.
Method (ours). Same as Method (b) but using our model (finetuned using metric learning with triplet loss) to compute image embeddings.

Looking at the results of Table 1. we can observe that our approach (consisting in weighting predictions based on the learned metric) was effective and allowed us to achieve an average score of 72.31, beating both proposed baseline methods (a) and (b).

5. Conclusions

Starting from the provided model, we implemented a metric learning approach based on triplet loss to learn a reliable metric between visual domains of the PACS dataset. We showed that enriching data with textual descriptions in natural language improves Domain Generalization. Specifically, the learned metric allows for weighting reasonably the predictions made by 3 different classifiers (trained separately on 3 source domains) on the previously unseen target domain. We achieved an average score of 72.31 improving on the provided baseline methods. Taking a look at the 2D projection of the embeddings we see that the learned metric is reasonable due to the fact that intuitive relationships between visual domains are observed.

Repository

Full code and data for the project are available in this github repository (follow README.md instructions to setup everything):

`https://github.com/pathselector-x/AML_Project`

References

[1] Da et al. Li. Deeper, broader and artier domain generalization. *ICCV*, 2017. 1

[2] Chenyun Wu. https://github.com/chenyunwu/describingtextures. 2

[3] Mikayla Timm Wu, Chenyun and Subhransu Maji. Describing textures using natural language. *ECCV*, 2020. 2