Domain Disentanglement through Natural Language

Milad Zakhireh, Seyedomid Mahdavi, Zohreh Lahijaniamiri
Politecnico di Torino
milad.zakhireh@studenti.polito.it
omid.mahdavi@studenti.polito.it
zohreh.lahijaniamiri@studenti.polito.it

Abstract

A fundamental challenge for machine learning models is generalizing to datasets of different distributions, one of the basic concepts of which is referred to as Domain Disentanglement. There have been many pieces of research work conducted touching on domain disentanglement, especially in medical imaging and computer vision areas. It aims to learn intrinsic representations of semantic concepts from captured images that can be generalized across domains. The task is quite challenging due to the presence of domain shift. In order to contribute to a better resolution of the problem, including ad-hoc language processing techniques can be highly beneficial. Learning directly from raw texts about images is a promising alternative that leverages a much broader source of supervision. Data and code are available at https://github.com/OmidMahdavii/Vision-language

1. Introduction

One of the severe constraints of deep models is their weakness in producing reliable and consistent results with any data passed at test time. This is mainly due to possible differences in the distribution of both training and test data, which is known as the problem of domain shift. One of the major effects of domain shift is that our estimates of the expected loss on the test set may be biased and cause a sharp drop in accuracy. Through a few steps, which will be discussed later in detail, we try to bridge this gap and assess the performance of the model.

Our goal is to train a model that works well on the target distribution. One of the techniques to improve the performance of the model is Domain Disentanglement. It is assumed that all domains have their own features by which they are characterized. There will be two types of features, domain-specific and category-specific. We will need to disentangle them both and then use the category-specific features to train the object classifier. We should also include a domain classifier to do the same using the domain-specific features. While the object classifier is developed based on a labeled set of source domain samples, the domain classifier uses an unlabeled set of target domain images along with that of samples from the source domain.

We can also explore the possibility of helping this process through the frozen Text Encoder of CLIP applying a textual description of the visual domain. We feed the CLIP Encoder with a visual attributes description corresponding to each training sample to improve the disentanglement of domain-specific features.

As a complementary task, we try fine-tuning the CLIP encoder with our own dataset. We develop a model unfreezing the last layer of the CLIP network. Afterwards, we replace CLIP with the aforementioned model and repeat the experiment.

2. Related Work

There have been several pieces of research work conducted all around the world contributing to the development of learning approaches. Domain adaptation approaches, in general, tend to learn a mapping between domains under such a condition that the target domain data are either fully labeled or have a few labeled samples. Ganin et al. [2] attempt to embed domain adaptation into the process of learning representation so that the final classification decisions are made based on features that are both discriminative and invariant to the change of domains. Their approach also attempts to match feature space distributions; however, this is accomplished by modifying the feature representation itself rather than by reweighing or geometric transformation. Thus, they focus on learning features that combine (i) discriminativeness and domain-invariance. Their paper proposes a new approach to domain adaptation of feed-forward neural networks, which allows large-scale training based on a large amount of annotated data in the source domain and a large amount of unannotated data in the target domain. On the other hand, the generalization of models learned on a single visual domain to novel domains has universally caught the attention of many researchers active in the field of object recognition. However, many of the current methods consider domain adaptation between a limited number of domains (usually one source domain and one target domain). As a technique to measure domain similarity is critical for domain adaptation performance, Peng et al. [4] propose a novel Domain2Vec model to provide vectorial representations of visual domains based on joint learning of feature disentanglement and Gram matrix. Formally, given N distinct domains $\hat{\mathcal{D}} = \left\{\hat{\mathcal{D}}_1, \hat{\mathcal{D}}_2, \ldots, \hat{\mathcal{D}}_N\right\}^{\dagger}$ domains, the aim is to learn a domain to vector mapping $\Phi : \hat{\mathcal{D}} \rightarrow V$, which is capable of predicting domain similarities that match their intuition about visual relations between different domains. The vectorization task is composed of two parts: a feature generator which computes the latent representations $f_\theta = \phi_\theta(x)$ of the input data, and a classifier which encodes the distribution $p(y \mid x)$ given the representation f_θ. They believe, using this novel model, relations between different domains can be learned and described more effectively. Thus, they carry out extensive experiments, both qualitatively and quantitatively, on the two benchmarks to demonstrate the advantage of their proposed model. To evaluate the effectiveness of their Domain2Vec model, they create two large-scale cross-domain benchmarks. The first one is TinyDA, which contains 54 domains and about one million MNIST-style images. The second benchmark is DomainBank, which is collected from 56 existing vision datasets. They test their model on the two datasets and achieve interesting results. In addition to the two previously introduced pieces of work, Radford et al. [5] demonstrate that the simple pre-training task of predicting which caption goes with which image is an efficient and scalable way to learn SOTA image representations from scratch on a dataset of 400 million (image, text) pairs collected from the internet. After pre-training, they use natural language to reference learned visual concepts or describe new ones enabling zero-shot transfer of the model to downstream tasks. While standard image models jointly train an image feature extractor and a linear classifier to predict some label, their model CLIP jointly trains an image encoder and a text encoder to predict the correct pairings of a batch of (image, text) training examples. They study the performance of their approach by benchmarking on over 30 different existing computer vision datasets. The model transfers non-trivially to most tasks and is often competitive with a fully supervised baseline. Inspired by the above-mentioned researches, we train our model based on the features extracted from the images followed by having their textual descriptions involved in the process to increase the domain-specificity of the features and improve model performance.

3. Domain Disentanglement through Natural Language

3.1. Baseline Model

The baseline model consists of training the Feature Extractor (ResNet-18), the Category Encoder (made up of some fully connected layers), and the Object Classifier (made up of one fully connected layer for category prediction) minimizing the Cross-Entropy Loss computed on the source domain. We train our model on a single domain and test it on the other domains separately. This model is implemented to track the potential improvements in the performances of our supplementary techniques, comparing the results.

3.2. Domain Disentanglement

To improve the performance of our model, considering Domain Adaptation setting can be very helpful. Its main goal is to reduce the domain distribution discrepancy between the source domain and the target domain so that the knowledge learned from the source domain can be further applied to the target domain [7]. In this step, we address this issue through Domain Disentanglement. The assumption is that every domain is characterized by domain-specific features f_{ds} and category-specific features f_{cs}. We first pass our data, including both the source domain and the target domain, to the feature extractor in order to compute the latent representations $f_\theta = \phi_\theta(x)$ of the input data. Now that the representations are computed, the encoders and classifiers come into play and contribute to the minimization of Cross-Entropy Loss for both the object classifier and the category classifier as the number of iterations grows. As a result, we will have a category classifier C trained with class labels to predict the class distributions and a domain classifier DC trained with domain labels to predict the domain distributions. In the second step, we remove domain information from f_{cs} and category information from f_{ds} through the cross-adversarial training step. This can be achieved by minimizing the negative entropy of the predicted domain distribution. Since some information might get lost in the feature disentanglement process, it cannot guarantee the information integrity in the feature disentanglement process, especially when the feature disentangler D consists of several fully connected and RELU layers. Thus, we also implement a feature reconstructor R to recover the original features f_G with the disentangled domain-specific features and category-specific features. With this technique, we should be able to see an improvement in the performance with respect to the baseline. The category encoder and domain encoder share the same architecture made up of three fully-connected and RELU layers. The reconstructor is composed of a single fully-connected layer which takes the concatenation of (f_{ds}, f_{cs}) pair as input and outputs a

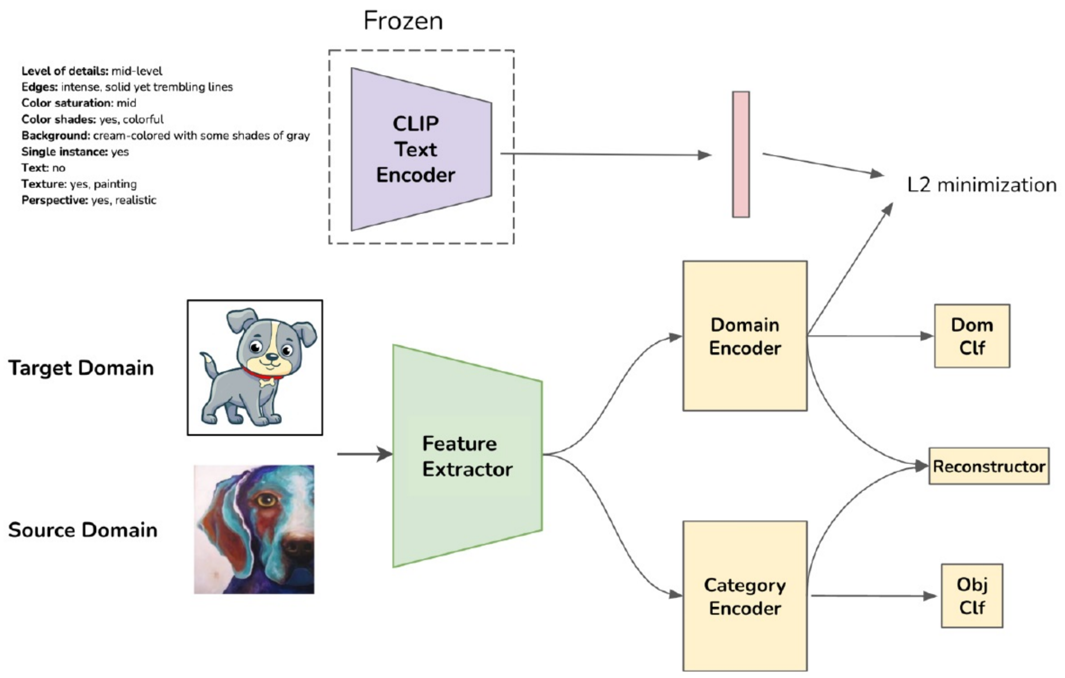


Figure 1. CLIP disentanglement components and architecture.

vector of the same size as f_θ. Denoting the reconstructed feature as $\hat{f}_G$, we can train the feature reconstruction process with the following loss:

$$\mathcal{L}_{rec} = \left\| \hat{f}_G - f_G \right\|_F^2 + KL\left(q\left(z \mid f_G\right) \| p(z)\right)$$

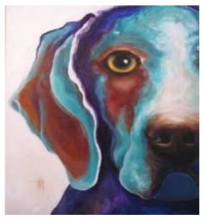
Where the first term aims at recovering the original features extracted by G, and the second term calculates Kullback-Leibler divergence which penalizes the deviation of latent features from the prior distribution $p\left(z_c\right)$ (as $z \sim \mathcal{N}(0, I)$).

3.3. CLIP Disentanglement

In order to be able to have the description of the images involved in our model's decision making, we first have to add a description to each image following particular guides e.g. mid-level is a description for the level of details as shown in Fig. 2, which is not relevant to the object. Following the above-mentioned instructions, we labeled 400 images of the dataset. We used the images that we have labeled together with previously labeled ones.

CLIP [5] is a neural network which efficiently learns visual concepts from natural language supervision. To be more specific, CLIP pre-trains an image encoder and a text encoder to predict which images were paired with which texts in the dataset.

In order to increase the domain specificity of the features to improve disentanglement, we aim to include the text in the disentanglement process. We pass the labels that we provided for the images to the frozen text encoder of CLIP and minimize an $L2$ loss between the features from the domain encoder and those from the CLIP text encoder. The architecture is illustrated in Fig. 1.



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

Figure 2. How to describe an image using features given.

3.4. CLIP Fine-tuning

Fine-tuning is a process that takes a model that has already been trained for a given task, and then tuning or tweaking that model to perform a second similar task. Since our dataset is different from CLIP's dataset, to make the model more task-specific, we fine-tune the CLIP weights instead of using the frozen network. There are two approaches to fine-tuning a model, unfreezing either the whole network or just some of the final layers. Updating all weights of the network is a quite time-consuming task as CLIP's architecture is huge. Thus, we freeze the entire network except for the last layer and update the gradient only for it. Ultimately,

Experiment	Total Loss Coefficients	Accuracy Cartoon	Photo	Sketch	Average
Baseline	-	59.04%	58.72%	94.07%	**70.61%**
Domain Disentanglement	$a = b = c = d = e = f = 1$	60.7%	45.05%	92.51%	66.09%
	$a = 4, b = c = d = e = 1, f = 0.25$	68.00%	60.07%	93.83%	73.97%
	$\boldsymbol{a = 4, b = c = d = e = 1, f = 0}$	65.36%	65.74%	94.43%	**75.18%**
CLIP Disentanglement	$a = 4, b = c = d = e = g = 1, f = 0.25$	65.7%	58.59%	94.37%	72.89%
	$\boldsymbol{a = 4, b = c = d = e = g = 1, f = 0}$	66.38%	67.07%	95.27%	**76.24%**
	$a = 4, b = c = d = e = 1, f = 0, g = 2$	56.83%	42.84%	89.94%	63.2%
CLIP Fine-tuning	$\boldsymbol{a = 4, b = c = d = e = g = 1, f = 0}$	65.27%	70.43%	94.55%	**76.75%**

Table 1. Accuracy for different target domains by building the model based on Art Painting with different loss coefficients.

we substitute the final model for the original CLIP model in the disentanglement task and compare the results.

4. Experiments

4.1. Datasets

To evaluate the performance of our model, we used PACS image dataset introduced by Li et al. [3]. 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. They created this new category by intersecting the classes found in Caltech256 (Photo), Sketchy (Photo, Sketch) [6], TU-Berlin (Sketch) [1] and Google Images (Art painting, Cartoon, Photo). This benchmark carries two important advancements: (i) it uniquely includes domains that are maximally distinct from each other, spanning a wide spectrum of visual abstraction, from photos that are the least abstract to human sketches which are the most abstract; (ii) it is more reflective of a real-world task where a target domain (such as sketch) is intrinsically sparse.

4.2. Baseline

In this project, we initialize our experiment series with a baseline in which the difference between domains is not taken into account. We simply train the model on a single domain (Art-Painting) and test it on others (Cartoon, Sketch, Photo). After testing the model on the target domains, the results we achieved for Cartoon and Sketch are not as satisfying as what we achieved for Photo. The baseline result for Photo is $\mathbf{94.07}\%$ while $\mathbf{59.04}\%$ and $\mathbf{58.72}\%$ for Cartoon and Sketch respectively (Table 1). From the results, it can be inferred that Photo has more in common with Art-painting than the other two domains, Sketch and Cartoon, regarding their visual characteristics. Hence, it is clear that additional improvements should be applied to the model in order to mitigate the problem.

4.3. Domain Disentanglement

In this experiment, we also add the target domain samples to the training set excluding their category labels because they are not going to be used for the object classifier. We apply our devised model to the PACS dataset. Considering several contributions in the final loss function, our aim is to minimize Cross-Entropy Loss for the object classifier L_{ce}^{class} and the domain classifier L_{ce}^{domain}. We should maximize Entropy Loss for the domain classifier considering the features extracted from the Category encoder L_{ent}^{domain}, and we also have to consider the maximization of the Entropy Loss for the object classifier considering the features extracted from the Domain Encoder L_{ent}^{class}. In the end, we calculate the Reconstruction loss as the sum of an $L2$ Loss L_{MSE}^{rec} and a KL Loss L_{KL}^{rec}. The total loss is computed as shown below:

$$\mathcal{L}_{tot} = aL_{ce}^{\text{class}} - bL_{ent}^{\text{class}} + cL_{ce}^{\text{domain}} - dL_{ent}^{\text{domain}} + eL_{MSE}^{\text{rec}} + fL_{KL}^{\text{rec}} \tag{1}$$

Since every loss in Eq. (1) is scaled differently, we have to consider unequal coefficients as hyper-parameters of our model for each one. Assigning all coefficients equal to $\mathbf{1}$, we obtain $\mathbf{22.7}\%$, $\mathbf{19.65}\%$, and $\mathbf{38.08}\%$ accuracy for Cartoon, Sketch, and Photo respectively which is not satisfying at all. Therefore, we implement several configurations for them and compare the outcome. For instance, we observe a significant improvement when we compute the average Entropy instead of Entropy. Thus, from now on, L_{ent} in Eq. (1) represents the average Entropy instead of Entropy i.e. Entropy divided by the number of samples.

In Table 1, different hyper-parameter sets and their corresponding results are depicted. Based on our observations, removing KL Loss from Eq. (1) produces a better performance. The best configuration for the domain disentanglement experiment resulted in an improvement of the average accuracy by $\mathbf{4.5}\%$ with respect to the baseline experiment. The configuration is as given below:

$$a = 4, b = 1, c = 1, d = 1, e = 1, f = 0$$

Experiment	Validation Accuracy Coefficients	Accuracy			
		Cartoon	Photo	Sketch	Average
Domain Disentanglement	α=1, β=0	65.36%	65.74%	94.43%	75.18%
	α=1, β=1	68.09%	66.02%	95.63%	**76.58%**
	α=2, β=1	63.1%	58.87%	93.95%	71.97%
	α=1, β=2	67.15%	50.11%	92.93%	70.06%
CLIP Disentanglement	α=1, β=0	66.38%	67.07%	95.27%	76.24%
	α=1, β=1	67.75%	69.2%	94.85%	**77.27%**
	α=2, β=1	63.48%	68.13%	94.25%	75.29%
	α=1, β=2	62.16%	64.49%	92.16%	72.94%
CLIP Fine-tuning	α=1, β=0	65.27%	70.43%	94.55%	76.75%
	α=1, β=1	66.72%	69.38%	94.61%	**76.9%**
	α=2, β=1	61.43%	61.98%	94.13%	72.51%
	α=1, β=2	67.32%	57.01%	90.54%	71.62%

Table 2. Accuracy for different target domains by building the model based on Art Painting using the best set of loss coefficients for each experiment with different validation accuracy coefficients.

During the validation step of the baseline experiment, we select the model with the best category classification accuracy ACC_{class} for the test step. However, in this experiment, we try including also the domain classification accuracy ACC_{domain} since the performance of the domain classification is important to be able to discriminate between different domains.

The final accuracy is calculated through the following equation:

$$\mathcal{ACC}_{tot} = \frac{\alpha ACC_{class} + \beta ACC_{domain}}{\alpha + \beta} \tag{2}$$

We train the model with the best loss configuration found in the previous stage to compute the accuracy in the validation step deploying different combinations of α and β which are illustrated in Table 2. During this experiment, by assigning both α and β to 1, we noticed a $\mathbf{1.5}\%$ improvement in the accuracy we had already achieved.

4.4. CLIP Disentanglement

To increase the domain specificity, we include the text in the disentanglement process using CLIP text encoder. We also consider the loss between the features extracted from the domain encoder and the output of the CLIP text encoder L_{clip} in the computation of total loss:

$$\begin{aligned}\mathcal{L}_{tot} = aL_{ce}^{\text{class}} - bL_{ent}^{\text{class}} + cL_{ce}^{\text{domain}} - dL_{ent}^{\text{domain}} \\ +eL_{MSE}^{\text{rec}} + fL_{KL}^{\text{rec}} + gL_{clip}\end{aligned} \tag{3}$$

For the CLIP disentanglement experiment the best set of hyper-parameters is reported below:

$$a = 4, b = 1, c = 1, d = 1, e = 1, f = 0, g = 1$$

Through implementing the mentioned weights, the average accuracy raises to $\mathbf{76.24}\%$ (Table 1). Moreover, adding the domain classifier accuracy to the validation step results in another $\mathbf{1}\%$ improvement in the average accuracy (Table 2).

4.5. CLIP Fine-tuning

The next potential solution to acquire better performance is to fine-tune the CLIP model. Due to the massive number of weights that CLIP includes, fine-tuning them all is a very expensive task. Therefore, we freeze all the CLIP layers except for the last one. Subsequently, we feed the network with the PACS images and their corresponding descriptions and train the model. After 200 iterations, the model which produces the minimum Cross-Entropy Loss is selected as our fine-tuned model.

The final step is to replace the CLIP text encoder in 4.4 with the fine-tuned one and repeat the training. For this experiment, only the best set of total loss coefficients from the CLIP disentangle experiment are evaluated. As Table 1 and Table 2 report, this technique did not affect our model as we expected and the average accuracy remained almost unchanged compared to the CLIP disentangle stage. Underfitting and the limitation of the number of CLIP fine-tuning iterations could be one of the possible reasons. Furthermore, the performance of CLIP highly depends on the provided descriptions. Assigning a more specific set of textual labels to the dataset images might improve the efficiency considerably.

5. Conclusion

In this paper, we examined several techniques to cope with the domain adaptation problem in object classification tasks. Our studies indicate that implementing domain disentanglement along with CLIP text encoder boosts the accuracy significantly with respect to the basic model, especially if the descriptions fed to the text encoder are precise enough. In addition, we proved that monitoring the domain classifier accuracy as well as the category classifier performance in the validation process leads to a better selection of the final model for the test stage. A possible solution to achieve an even more reliable accuracy is to apply Domain Generalization (DG) technique in the experiment. This way, the model is trained based on several source domains instead of one, and it is tested on a target domain never seen during training.

References

[1] Mathias Eitz, James Hays, and Marc Alexa. How do humans sketch objects? *ACM Transactions on graphics (TOG)*, 31(4):1–10, 2012. 4

[2] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The journal of machine learning research*, 17(1):2096–2030, 2016. 1

[3] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M Hospedales. Deeper, broader and artier domain generalization. In *Proceedings of the IEEE international conference on computer vision*, pages 5542–5550, 2017. 4

[4] Xingchao Peng, Yichen Li, and Kate Saenko. Domain2vec: Domain embedding for unsupervised domain adaptation. In *European conference on computer vision*, pages 756–774. Springer, 2020. 2

[5] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR, 2021. 2, 3

[6] Patsorn Sangkloy, Nathan Burnell, Cusuh Ham, and James Hays. The sketchy database: learning to retrieve badly drawn bunnies. *ACM Transactions on Graphics (TOG)*, 35(4):1–12, 2016. 4

[7] Wen Xu, Jing He, and Yanfeng Shu. Transfer learning and deep domain adaptation. *Advances and Applications in Deep Learning*, page 45, 2020. 2