Using self-supervised learning to improve performance on domain adaptation

Ali Dadsetan
dadsetan.ali@gmail.com

1 Introduction

Given a collection of labeled data drawn from a source domain with a specific distribution, the goal of domain adaptation is to make predictions over a target domain, which has a different but related distribution to the source domain and from which we have only access to unlabeled data. In other words, the labeled data in the target domain can only be used in the evaluation phase. For example, when we have access to many images and their categories downloaded from the internet for training, but we want to predict categories of images taken professionally with expensive cameras and controlled lighting conditions, the problem can be considered a domain adaptation problem. This could happen because of not having access to enough samples taken in that professional setting, whereas we need to be able to categorize them correctly. The dataset we are using in this project is even more challenging.

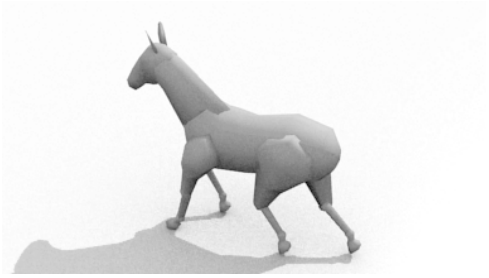
A simple approach for this task is to train a linear classifier over the representation of images sampled from the source domain by applying some feature-extractor on the images. Then we can use this linear classifier and the same feature-extractor to predict labels on the target domain. However, this approach does not leverage the unlabeled target data points sampled from the target domain.

In this project, we explore the possible benefits of using a joint dataset combining the unlabeled target samples and the labeled source samples for training the feature-extractor in a self-supervised fashion. We highlight that the performance of the linear classifier will increase when feature-extractor is trained in this way. This is numerically validated on the VisDA2017 dataset, where we show that the multi-class accuracy increases from 51% to 66%.

2 Method

The overall strategy is to find a shared representation between the source and target domain, with the property that when some algorithm is trained on the representations of the source domain, it will generalize naturally to the representations of the target domain. In other words, we want to improve the fitness of the feature-extractor for the domain adaptation task. For simplicity, we would start with a pre-trained Resnet50 architecture as the feature-extractor, and use a modified version of SimCLR, a self-supervised learning algorithm, to improve on the mentioned property. It is worth noting that the Visda2017 dataset is a challenging dataset, with significant differences between the distribution of source and target domain (figure 1). In the following sub-sections, we first give a brief review of the SimCLR paper Chen et al. (2020) in 2.1, and then we give more details about our algorithm and the training of the linear classifier and the feature-extractor in 2.2. Then we will explain our modifications to the SimCLR algorithm in sub-section 2.3.

(a) A bicycle in source domain.



(b) A horse in source domain.

(c) A bicycle in target domain.

(d) A horse in target domain.

Figure 1: Comparison between Visda2017's source and target domains.

2.1 SimCLR

The SimCLR method, Chen et al. (2020), is a self-supervised way to find representations that are invariant to some specific transformations (augmentations). The network consists of four different parts: the augmentation, the encoder, the projection layer, and the contrastive loss function. These components are shown in figure 2. In this figure, for each image x, the algorithm applies two different transformations, t and t', from a set of predefined transformations, $\mathcal{T}$, to get augmented images $\tilde{x}_i$ and $\tilde{x}_j$. In that paper, the authors showed that $\mathcal{T}$ should at least contain random cropping and random changes in color. After that, the augmented images go through some encoding part (the feature-extractor in our case), $f(.)$, to get $\boldsymbol{h}_i$ and $\boldsymbol{h}_j$. And after that they go through some MLP named $g(.)$ in the figure, which reduces the dimension of $\boldsymbol{h}_i$ and $\boldsymbol{h}_j$ to get $\boldsymbol{z}_i$ and $\boldsymbol{z}_j$. Then the loss function comes into play. It will make the $\boldsymbol{z}_i$ and $\boldsymbol{z}_j$ closer to each other. We will discuss the loss function in more detail in 2.1.1

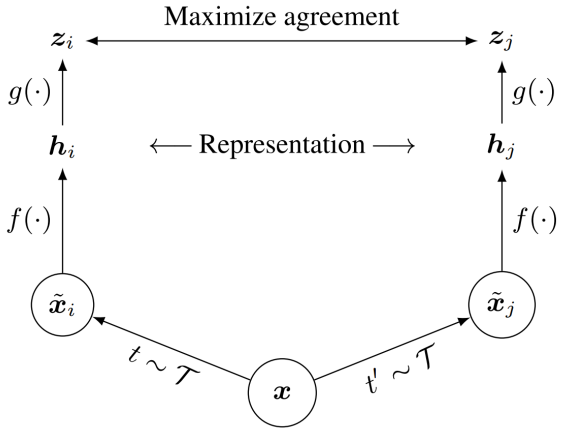


Figure 2: Components of SimCLR algorithm (from Chen et al. (2020)).

2.1.1 Loss function

Let us review the loss function used in the original SimCLR paper in more detail. For any given batch with n samples, $x_1, \ldots, x_n$, we could consider the representations after the projection layer, $z_1, \ldots, z_n$, and representations of an augmented copy, $z_{n+1}, \ldots, z_{2n}$. In this setting, for each i, $1 \leq i \leq n$, the SimCLR paper has defined the ℓ_{ij} by

$$\ell_{i,j} = -\log \frac{\exp(\text{sim}(z_i, z_j)/\tau}{\sum_{k \neq i} \exp(\text{sim}(z_i, z_k)/\tau)} \tag{1}$$

This is like doing a softmax over the similarities of the pairs of z_i and z_j. So optimizing a loss function like that would make z_i closer to z_j and farther from other z_ks. Then, the SimCLR paper defined the total loss function of the batch by the summation $L = \sum_{i=1,\ldots,n} \ell_{i,n+i}$, making z_i's more similar to z_{n+i}'s (their augmented counterpart), and less similar to other z_ks.

2.2 The overall algorithm

We will give more details about the algorithm. We have two different training processes, which we define here distinctly. One training process is done on the linear classifier in a supervised way, and the other training is done on the feature-extractor in a self-supervised fashion.

Linear Classifier Training The linear classifier is trained on the outputs of the feature-extractor over the source domain images and their labels. In this process, no changes to the weights of the feature-extractor are made. Then, to evaluate the model's accuracy, we would first apply the feature-extractor on the images of the target domain and then apply the linear-classifier on the obtained representations to predict a class for each image. Finally, we compare predicted classes with the target domain images' labels to report the model's accuracy. We emphasize again that we do not use any of the labels of the target domain in any training process. We call this accuracy the accuracy of the linear classifier (over the target domain). This training is quite naive, and we expect it to have poor results in such challenging problems. This linear classifier is chosen to show a proof of concept for using self-supervised learning to solve domain adaptation problems. Next, we are going to discuss the self-supervised learning part.

Feature-extractor training Training of the feature-extractor happens in a self-supervised way. This self-supervised algorithm is a modification of the SimCLR algorithm. We will discuss these modifications in 2.3. The modified version still has the four different parts we mentioned in 2.1. We put our feature-extractor as $f(.)$ in the SimCLR setting and trained over a combined dataset of both target and source samples. We take advantage of the labels of the source domain here. After that, we drop the MLP, $g(.)$, and use the trained feature-extractor, $f(.)$, for the training of the linear classifier explained above. We propose that in this way, the accuracy of a linear classifier trained with the new feature-extractor will be better compared to a linear classifier trained with the old feature-extractor. This is because the new feature-extractor gives a better representation of both the source and the target domain.

2.3 Proposed Modifications to SimCLR

First, there was some modification in the augmentation process. Most notably, for the images of the source domain, rendered images from different angles and with different lighting were treated as different augmentations of the same object. This is to create the feature-extractor invariant to 3D rotations available in the Visda2017 dataset. We believe in this way the feature-extractor will gain some insight into the 3D nature of the objects, which will help it in the domain adaptation task. A similar idea has been used in Azizi et al. (2021) in the context of medical imaging. We refer to Section 2.3.1 for more details on data augmentation.

Second, the loss function of the SimCLR algorithm has been changed to consider the labels whenever it is allowed to use them. A similar approach has been proposed in Khosla et al. (2020). Remarkably, both methods use labels to find representations that yield close features for images in the same category. However, Khosla et al. (2020) needs to address the limitation of not having access to target labels during training. This is essential because a shared representation of both domains is required for generalizing learning from the source domain to the target domain. More details about the loss function are given in 2.3.2.

2.3.1 Augmentation and more details about dataset

The VisdDA2017, Peng et al. (2017), source domain dataset has 150k images in 12 different categories, which are 2D renderings of only 1900 different synthetic 3D objects (i.e., Each model has 80 different renderings). The target domain has around 50k images in the same categories. The 80 different renderings of the same model are treated as transformations of the same object, so the size of the source dataset effectively will be 1900. For having a balanced number of source and target samples in the self-supervised training dataset, the elements of the source dataset are repeated ten times. Every time an element of this dataset is fetched, a random image of the set of 80 renderings of the object is returned. Furthermore, an additional augmentation in the source domain is done, adding random colors to the original gray-scale image.[1]

2.3.2 Changes to the loss function

Reasons for the Changes made to the original loss function are discussed at the start of this section (2.3). The idea is to change the denominator for ℓ_{ij}, equation 1, so that it would penalize distance between objects of the same category more. Also, it should not use the labels of the target domain. This could be done by simply giving different coefficients to different components of the summation in the denominator.

More formally, by defining p_{same} as the penalty coefficient for objects in the same category, and $p_{\text{different}}$ as the penalty for objects with different categories, the below definition for ℓ'_{ij} would take care of both of our criteria.

$$\ell'_{i,j} = -\log \frac{\exp(\text{sim}(z_i, z_j)/\tau)}{\sum_{k \neq i} p_k \exp(\text{sim}(z_i, z_k)/\tau)}$$

where p_k is the function

$$p_k = \begin{cases} 1 & x_i \notin \text{source or } x_j \notin \text{source} \\ p_{\text{same}} & y_k = y_i \\ p_{\text{different}} & y_k \neq y_i \end{cases}$$

and y_i is the category (label) of x_i. In the experiment leading to the result of this report, p_{same} was 0.1, and $p_{\text{different}}$ was 10. [2]

3 Results

After each epoch of self-supervised training over the joint dataset described in subsection 2.3.1, the projection layer of the SimCLR network, $g(.)$, is removed (temporarily) and replaced by a linear classifier, with weights of the linear classifier randomly initiated. Then after freezing the encoder part, we would train the linear layer over 1% of the source domain labeled data, and evaluate the multi-class accuracy

[1] Implementation of this section can be found at https://github.com/alidadsetan/SimCLR-VisDA/tree/main/dataset

[2] Implementation of this section can be found at https://github.com/alidadsetan/SimCLR-VisDA/blob/209cc4a378ac158d4cbded27df6376268b14e201/SimCLR.py#L180

of the resulting network on all elements of the target domain. This accuracy is reported in Figure as the `adaptation_acc_one_percent`. After that the projection layer is re-attached and the self-supervised part is continued.

We can see in Figure 3 that the lesser the loss function of the self-supervised algorithm becomes, the more accuracy is achieved on the target domain after training the linear classifier. This validates the stated hypothesis in section 1. This relation is not obvious and does not hold automatically, as the `train_loss_epoch` is the loss for the self-supervised training, described in 2.2, and the `adaptation_acc_one_percent` is the accuracy of linear-classifier over the representations of the target domain by applying feature-extractor, also described in 2.2.

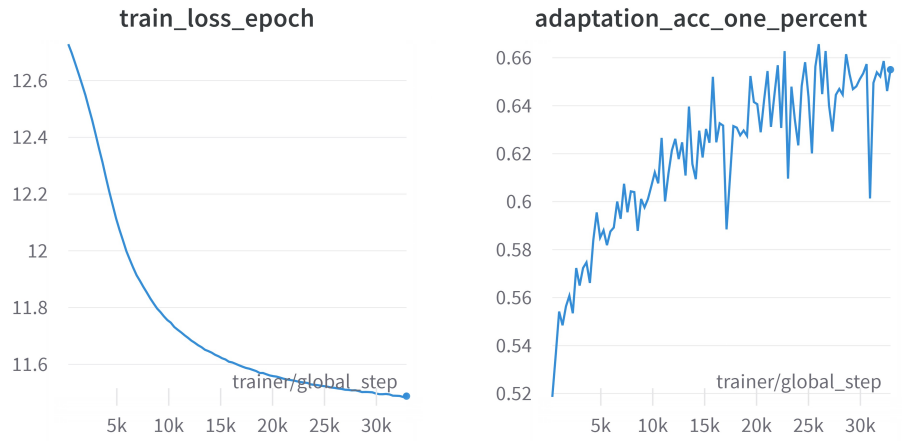


Figure 3: Loss function for self-supervised learning and accuracy of the linear classifier.

4 Future Directions

There are multiple hyper-parameters that could be fine-tuned in this project. For example, the p_{same} and $p_{\text{different}}$, 2.3.2, the amount of repeating in 2.3.1, which is ten in this project, and the batch size for training the SimCLR algorithm can be fine-tuned. Also, as our approach is agnostic to the model architecture of the feature-extractor, we could try different feature-extractors instead of ResNet50.

However, considering that the linear classifier is the most simple method of classification, we could ask if the resulting feature-extractor in this project would also improve the performance of other more sophisticated algorithms. This can be investigated with other methods which use CNN architectures in them. As candidate methods, we will explore the possibility of combining our method with those in Na et al. (2021) and Hoyer et al. (2022). We believe that this approach is a promising way to improve state of the art on the problem.

Acknowledgement

I want to thank the ML collective community, as this project was possible because of their computing support. Also, I want to thank Mr. Vivek Natarajan, who mentored me in this project.

References

Azizi, S., Mustafa, B., Ryan, F., Beaver, Z., Freyberg, J., Deaton, J., Loh, A., Karthikesalingam, A., Kornblith, S., Chen, T., et al. (2021). Big self-supervised models advance medical image classification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3478–3488.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. (2020). A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR.

Hoyer, L., Dai, D., Wang, H., and Van Gool, L. (2022). Mic: Masked image consistency for context-enhanced domain adaptation. *arXiv preprint arXiv:2212.01322*.

Khosla, P., Teterwak, P., Wang, C., Sarna, A., Tian, Y., Isola, P., Maschinot, A., Liu, C., and Krishnan, D. (2020). Supervised contrastive learning. *Advances in Neural Information Processing Systems*, 33:18661–18673.

Na, J., Jung, H., Chang, H. J., and Hwang, W. (2021). Fixbi: Bridging domain spaces for unsupervised domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1094–1103.

Peng, X., Usman, B., Kaushik, N., Hoffman, J., Wang, D., and Saenko, K. (2017). Visda: The visual domain adaptation challenge.