Deep Domain Adaptation

Homework 3

Ludovisi Linda
s278221

June 2020

Academic Year 2019/2020

1 Homework Description

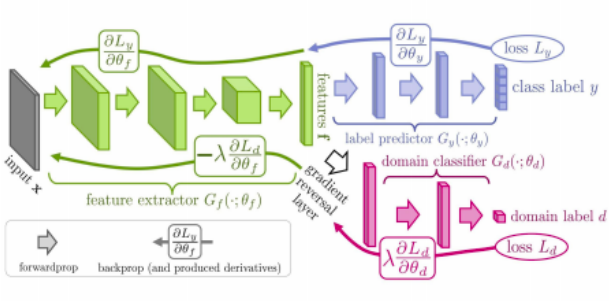
The aim of the project is to implement DANN, a Domain Adaptation algorithm, performed on the PACS dataset using the AlexNet model. The Domain Adaptation learning approach is suited for a scenario in which data at training and test time come from similar but different distributions. In fact, for domain transfer to be achieved, predictions must be made based on features that cannot discriminate between the training and test domains.

2 The Dataset

The dataset used in this project is the PACS dataset, which contains images belonging to Art painting, Cartoon, Photo and Sketch domains, in a way that each domain has 7 classes. In order to load the dataset, the *ImageFolder* class provided by PyTorch was used: every different domain was loaded as a different dataset, specifying *train* and *eval* transformations. Each image was preprocessed with a *CenterCrop* 224×224 in order to fit AlexNet model and then normalized with mean and variance of the ImageNet dataset.
In our scenario, the source dataset is Photo, while the target dataset is Art.

3 Implementing the model

DANN was implemented in PyTorch using an AlexNet-like architecture: the convolutional layers of AlexNet are intended as the feature extractor layers of the net (from now on we will call them G_f); the last FC layers are used for classification purposes (we will refer to them with G_y), they have 7 output neurons; and finally a separate branch with the same architecture of AlexNet fully connected layers will be called as G_d: it is very similar to G_y with the only difference that it has only 2 output neurons. In fact, it will be used to discriminate between the two domains.



In order to improve the performance of this classification task, the ImageNet weights were loaded into the feature extractor branch G_f, while the weights of the original classifier were copied into both G_y and G_d.
Also, in order to implement the DANN logic, the *forward function* of standard AlexNet was

changed, implementing a flag *alpha* which selects the correct branch to forward the data in, in a way that:

- if the data are forwarded to G_y, the gradients are computed as always;
- if the data are forwarded to G_d, the gradients are reverted with the Gradient Reversal layer.

Implementing the forward in this way leads to the phenomenon that, as the training progresses, the approach promotes the emergence of features that are **discriminative** for the main learning task on the source domain and **indiscriminate** with respect to the shift between the domains.

4 Domain Adaptation

The goal of this homework is to classify the target domain Art Painting, given the source domain Photo.

4.1 Training without validating

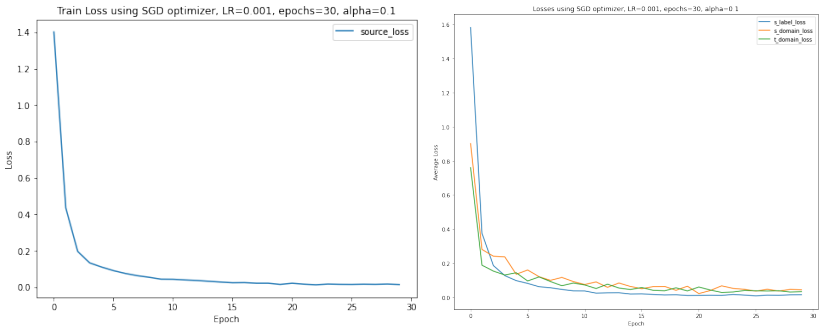
In the following section a training without and with DANN will be performed, in order to see how a training with/without DANN behave.
First, the training without adaptation was performed: the training dataset was Photo, while the test dataset was Art Painting. In particular, SGD optimizer was used, with a $momentum = 0.9$ and a $weight\_decay = 0.00005$.
Then, the training with adaptation was performed: at each epoch, an iteration over the batches of the target dataloader was first done (target dataset is longer than source dataset) and a separate dataloader that iterates over the source dataset was used. To account for the fact that source dataset is smaller than the target dataset, the raised exception was handled with a try/except block.
So a single training iteration was divided in three steps that were executed sequentially before calling optimizer.step():

- the model was first trained on source labels by forwarding source data to G_y, updating the gradients with *loss.backward()*.
- the discriminator was then trained by forwarding source data to G_d (labels are 0) and gradients were updated.
- finally, the discriminator was trained by forwarding target data to G_d (labels are 1) and gradients were updated.

The parameters used in these experiments were :

Epochs	**LR**	**alpha**	**Weight Decay**
30	0.001	0.1	0.00005

Test accuracy without DANN was : **0.4711**, while test accuracy with DANN was: **0.4887**.
The following plots show the trend of classification loss (source dataset) and the one of the loss of the domain classifier (source and target dataset).



As we can see, the first figure does not show any domain loss: it is because it is referred to the training without DANN. There, the loss seem to decrease as the number of epochs increases, showing a little overfitting in the last epochs.
The second plot shows the behavior of the loss of the classifier and the one of the domain classifiers: while the classification loss has a trend that decreases with the increasing of the number of epochs, the domain classifier losses have a trend that increases/decreases with the number of epochs. This could be explained understanding how the DANN works: it is trained to discriminate between source and target, while the feature extractor layers are generating domain invariant features.

5 Cross Domain Validation

Validation is an open problem in domain adaptation, since the goal of this task is to build a classifier which is robust w.r.t. samples that belong to a different domain. In order to perform validation, a training with and without DANN was performed on two other datasets: their domain was *Sketch* and *Cartoon*.

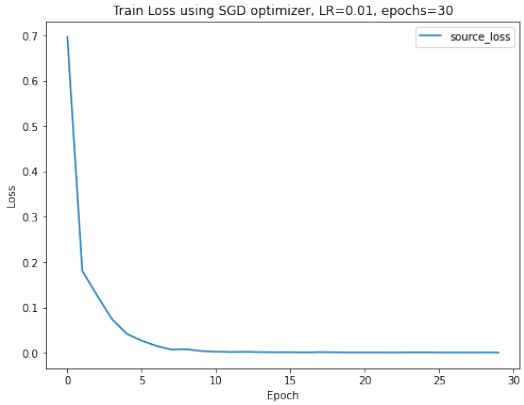
5.1 Training with validation, without DANN

First, a training on source dataset (Photo dataset) was performed with different hyperparameter configurations and tested on Cartoon and Sketch dataset. Final accuracies were averaged, in order to obtain the best overall configuration.

Epochs	**LR**	**Weight Decay**	**Acc. Sketch**	**Acc. Cartoon**	**Mean Acc.**
30	0.01	0.00005	60.98	32.42	**46.70**
30	0.001	0.00005	42.23	25.19	33.71
30	0.0001	0.00005	49.31	30.37	39.84
40	0.01	0.00005	52.73	31.29	42.01
40	0.001	0.00005	59.22	28.71	43.96
40	0.0001	0.00005	48.38	24.80	36.59
20	0.01	0.00005	62.64	28.22	45.76
20	0.001	0.00005	38.33	29.24	33.78
20	0.0001	0.00005	36.37	23.43	29.90

Table 1: Accuracy on Cartoon, Sketch dataset (without DANN)

As we can see from Table 1, the best hyperparameter configuration was the one with $Epochs = 30$, $LR = 0.01$, $weight\_decay = 0.00005$ with an avg. accuracy of **46.70**.
This model was also tested on Art dataset, in order to see how it behaves. Final accuracy was **48.58**.



The plot shows the trend of the training loss as the number of epochs increases. Here, the loss reaches a minimum (around epoch 15) and then plateaus. The final accuracy is better than the one obtained in Section 4.1. Also, it may perform well on different domains because it was validated on two different datasets.

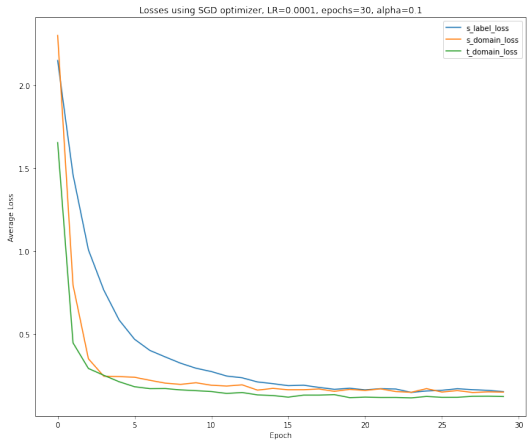
5.2 Training with validation, with DANN

Then, a training with DANN on source dataset (Photo dataset) was performed with different hyperparameter configurations and tested on Cartoon and Sketch dataset. Final accuracies were averaged, in order to obtain the best overall configuration.

alpha	Epochs	LR	W. Decay	Acc. Sketch	Acc. Cartoon	Mean Acc.
0.1	30	0.001	0.00005	58.34	28.95	43.64
0.1	30	0.0001	0.00005	54.34	35.98	**45.16**
0.01	30	0.01	0.00005	45.94	35.59	40.76
0.01	30	0.001	0.00005	52.29	29.10	40.69
0.01	30	0.0001	0.00005	38.91	33.98	36.44
0.001	30	0.01	0.00005	57.86	29.24	43.55
0.001	30	0.001	0.00005	49.95	27.58	38.76
0.001	30	0.0001	0.00005	34.47	24.36	29.41

Table 2: Accuracy on Cartoon, Sketch dataset (with DANN)

As we can see from Table 2, the best hyperparameter configuration was the one with $alpha = 0.1$, $Epochs = 30$, $LR = 0.0001$, $weight\_decay = 0.00005$ with an avg. accuracy of **45.16**.
This model was also tested on Art dataset, in order to see how it behaves. Final test accuracy was: **48.14**.



The plot shows the trend of the classification loss and the trend of the domain losses, performed on a training with Photo and Art datasets.
Even if the final accuracy is smaller, compared with the one obtained in Section 4.1, the model is more reliable because it is optimized with hyperparameters that were tested and therefore evaluated on two different domains.