POLITECNICO
DI TORINO

Report Homework 3
Course: Machine learning and deep learning

Davide Bussone

16 June 2020

Abstract

The aim is to address image classification task also for pictures having a different distribution. This means that in an environment composed of all equal labels, the domain of data exploited to train the model is different from the one where it is evaluated. So, the proposed architecture tries to let the network learn how to discriminate between the domains in view to classify correctly many images as possible.

1 The Dataset

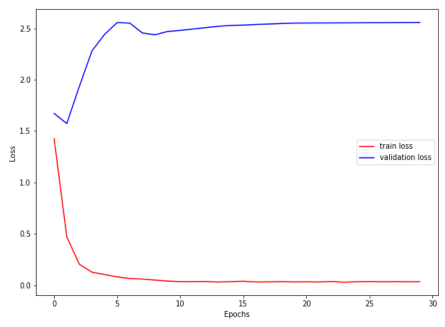
The studied dataset is called PACS. As name suggests, it is composed by pictures subdivided in 4 different domains (Photo, Art painting, Cartoon, Sketch); the differences between these subsections is highlighted in figure 4. Each domain has the same 7 different categories, as portrayed in image 3. Each folder's name in PACS correspond to a domain. As training dataset, "Photo" path's pictures are used, while as testing set "Art painting" folder is chosen. In the final part, both "Sketch" and "Cartoon"'s images are considered in view to validate the model on more domains. As a consequence, the network may be able to generalize better. Each input of the network is resized to 256, center cropped at 224 and normalized by using ImageNet's mean and standard deviation.

2 Contribution of domain adaptation

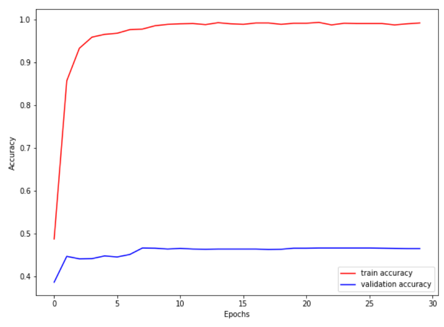
At the beginning a simple classification task is performed without domain adaptation. As said in section 1, images from "Photo"'s folder populate the train dataloader, while the output model will be tested, directly, on "Art painting"'s pictures. The optimizer used is stochastic gradient descent, with momentum 0.9, and in table 1 training details are shown. In addition, also the accuracy score is represented. As images 1a and 1b infer, the score is definitively lower compared to training found.

Learning Rate	Decay rate	Weight Decay	Number of epochs	Step size	Test accuracy
$1e^{-3}$	0.1	$5e^{-5}$	30	10	46.41%

Table 1: Training details and test accuracy without domain adaptation



(a) Loss behaviour without domain adaptation

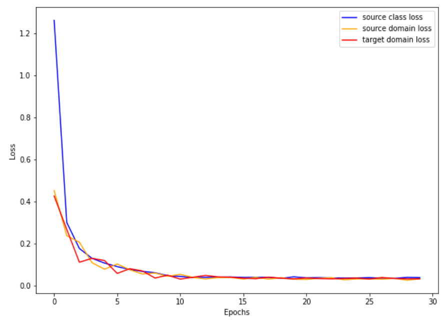


(b) Accuracy behaviour without domain adaptation

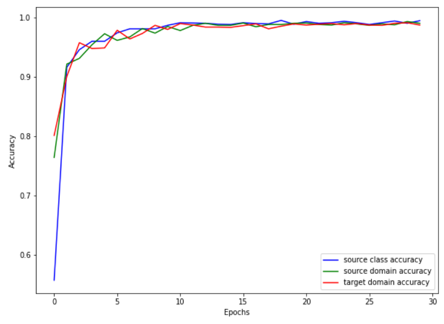
Then, the domain adaptation (DANN) model is built by modifying the structure of Alexnet. The convolutional layers are always the same, but the classificator has two branches. The first one is fit to predict the correct targets, while the second is able to recognize the two different domains. This last object shares the same structure, biases and weights with the other, the only difference is the ending, in fact it has two nodes ("Photo" or "Art painting") rather than 7. Also the backpropagation changes a bit, it concerns an additional hyperparameter, paper [1] calls it alpha and it represents the weight of the reversed backpropagation. It is the number to multiply to the reversal gradient. So, talking about the network's forward method, if alpha is given, it trains the domain classifier along the feature extractor block and it computes the gradient reverse layer, on the other case it performs the label classification. The training phase involves three different steps. Firstly, the net is fit to classify the targets of the source dataloader. Secondly, again the train dataloader is iterated, but now the labels turn into the domains. Finally, the network is trained on the target dataloader with the aim to recognize the correct domain. Board 2 shows the training details and the accuracy score reached on the test set. Lineplots 2a and 2b portrays the behaviour of the network during the steps both on source and target dataloader.

Learning Rate	Decay rate	Weight Decay	Number of epochs	Step size	Alpha	Test accuracy
$1e^{-3}$	0.1	$5e^{-5}$	30	10	0.1	41.60%

Table 2: Training details and test accuracy with domain adaptation



(a) Loss behaviour with domain adaptation



(b) Accuracy behaviour with domain adaptation

3 Hyperparameters optimization on art painting

In this phase, "Art painting" subset is exploited both as validation and test set. The selected hyperparameter search method is the random search, with a total of 8 iterations. It consists of selecting a multinoulli distribution for discrete hyperparameters or an uniform distribution for the positive-real valued ones. Random search, in this case, is preferred to grid search because it finds good solutions faster. This is due to the fact that random search has no wasted experimental runs, unlike the grid when two value of a hyperparameter (given values of the other ones) would give the same result. As bench 5 infers, the chosen hyperparameters, for the experiment without domain adaptation, are the learning rate, the decay factor and the weight decay. As table 4 portrays, with domain adaptation, the alpha's value is also optimized. The best combination is, finally, evaluated again on "art painting", which acts as test set, and the results are represented in board **??**.

Iteration	Learning Rate	Decay rate	Weight Decay	Accuracy
1	$1e^{-5}$	0.28	$5e^{-6}$	24.12%
2	$1e^{-3}$	0.13	$5e^{-5}$	48.13%
3	$1e^{-5}$	0.48	$5e^{-4}$	29.31%
4	$1e^{-3}$	0.35	$5e^{-6}$	50.28%
5	$1e^{-3}$	0.39	$5e^{-6}$	49.01%
6	$1e^{-4}$	0.47	$5e^{-4}$	43.45%
7	$1e^{-4}$	0.38	$5e^{-5}$	43.24%
8	$1e^{-5}$	0.34	$5e^{-5}$	25.07%

Table 3: Random search iterations without domain adaptation, considering art painting as validation set

Iteration	Learning Rate	Decay rate	Weight Decay	Alpha	Accuracy
1	$1e^{-5}$	0.42	$5e^{-6}$	0.47	27.40%
2	$1e^{-4}$	0.32	$5e^{-4}$	0.12	40.12%
3	$1e^{-4}$	0.14	$5e^{-5}$	0.29	40.73%
4	$1e^{-4}$	0.44	$5e^{-4}$	0.17	42.32%
5	$1e^{-4}$	0.38	$5e^{-4}$	0.34	45.26%
6	$1e^{-5}$	0.50	$5e^{-4}$	0.03	29.92%
7	$1e^{-4}$	0.46	$5e^{-4}$	0.04	40.53%
8	$1e^{-3}$	0.32	$5e^{-6}$	0.30	45.61%

Table 4: Random search iterations with domain adaptation, considering art painting as validation set

Experiment	Learning Rate	Decay rate	Weight Decay	Alpha	Test Accuracy
No Dann	$1e^{-3}$	0.35	$5e^{-6}$	NaN	45.36%
Dann	$1e^{-3}$	0.32	$5e^{-6}$	0.30	45.41%

Table 5: Best accuracy results, for both experiments, tested on Art painting subset

4 Hyperparameters optimization on other dataloaders

In view to validate "Photo" to "Art painting" subset, without cheating like in section 3, the performances both for source dataloader to "Cartoon" subset and for train dataloader to "Sketch" folder are measured again through a random search, already described in section 3. So, this approach is applied on Photo to Cartoon and on Photo to Sketch, both with and without domain adaptation. In addition, the accuracy scores are averaged for each set of hyperparameters, in this way the contribution of "Cartoon" and "Sketch" has the same weight in the average computation. Tables 6 and 7 show the obtained results. The best set of hyperparameters is then evaluated on "Art painting" subset, as board 8 infers. In this case there are no improvements with domain adaptation, but it is quite normal, due to the smallness of PACS dataset.

Learning Rate	Decay rate	Weight Decay	Cartoon Accuracy	Sketch Accuracy	Average accuracy
$1e^{-3}$	0.27	$5e^{-6}$	25%	43%	34%
$1e^{-4}$	0.21	$5e^{-5}$	23%	48%	35.5%
$1e^{-3}$	0.23	$5e^{-4}$	25%	47%	36%
$1e^{-4}$	0.18	$5e^{-4}$	22%	32%	27%
$1e^{-4}$	0.37	$5e^{-6}$	28%	52%	40%
$1e^{-3}$	0.37	$5e^{-6}$	30%	46%	38%
$1e^{-4}$	0.21	$5e^{-5}$	22%	49%	35.5%
$1e^{-3}$	0.15	$5e^{-5}$	27%	61%	44%

Table 6: Random search iterations without domain adaptation, considering an average between cartoon and sketch as validation sets

Learning Rate	Decay rate	Weight Decay	Alpha	Cartoon Accuracy	Sketch Accuracy	Average accuracy
$1e^{-3}$	0.45	$5e^{-5}$	0.06	26%	17%	21.5%
$1e^{-2}$	0.19	$5e^{-6}$	0.12	17%	16%	16.5%
$1e^{-4}$	0.39	$5e^{-6}$	0.18	16%	26%	21%
$1e^{-3}$	0.46	$5e^{-4}$	0.29	19%	32%	25.5%
$1e^{-4}$	0.38	$5e^{-6}$	0.23	22%	21%	21.5%
$1e^{-3}$	0.10	$5e^{-6}$	0.14	25%	32%	28.5%
$1e^{-2}$	0.26	$5e^{-6}$	0.26	17%	21%	19%
$1e^{-2}$	0.39	$5e^{-5}$	0.12	17%	20%	18.5%

Table 7: Random search iterations with domain adaptation, considering an average between cartoon and sketch as validation sets

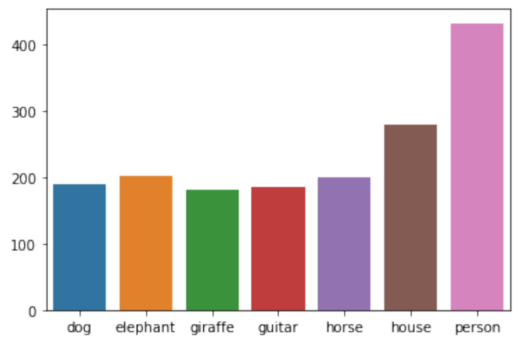
Learning Rate	Decay rate	Weight Decay	Alpha	Test accuracy
$1e^{-3}$	0.15	$5e^{-5}$	0.12	48.73%
$1e^{-3}$	0.10	$5e^{-6}$	0.14	43.13%

Table 8: Best hyperparameters set evaluated on art painting subset

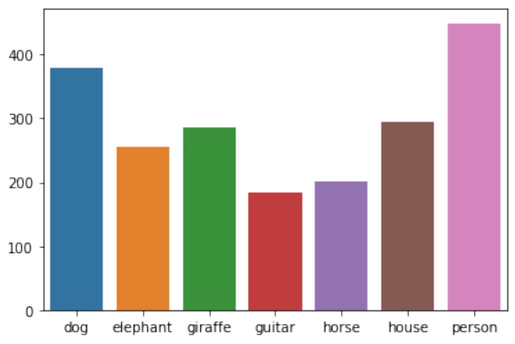
References

[1] Y.Ganin and V.Lempitsky. *Unsupervised domain adaptation by backpropagation.*

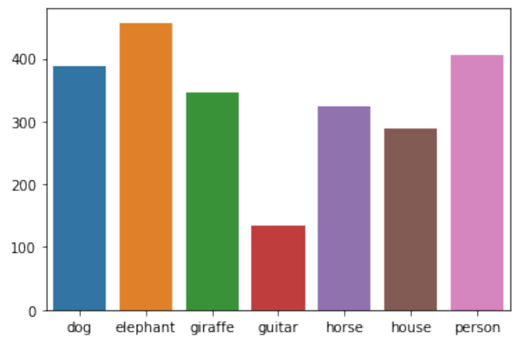
Appendix



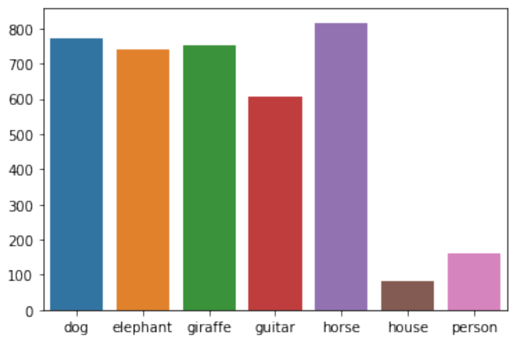
(a) Photo subset



(b) Art painting subset

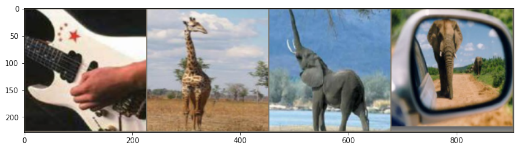


(c) Cartoon subset

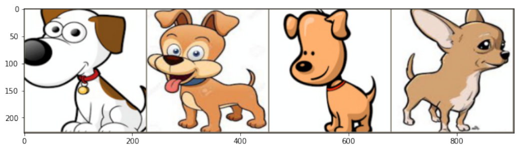


(d) Sketch subset

Figure 3: Target classes for each domain


(a) Photo subset

(b) Art painting subset


(c) Cartoon subset

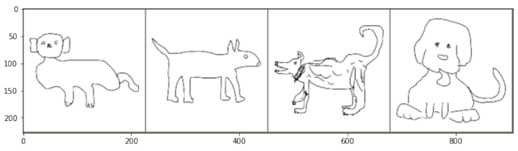

(d) Sketch subset

Figure 4: Sample images for each domain