Machine Learning and Deep Learning

Report - Homework 3

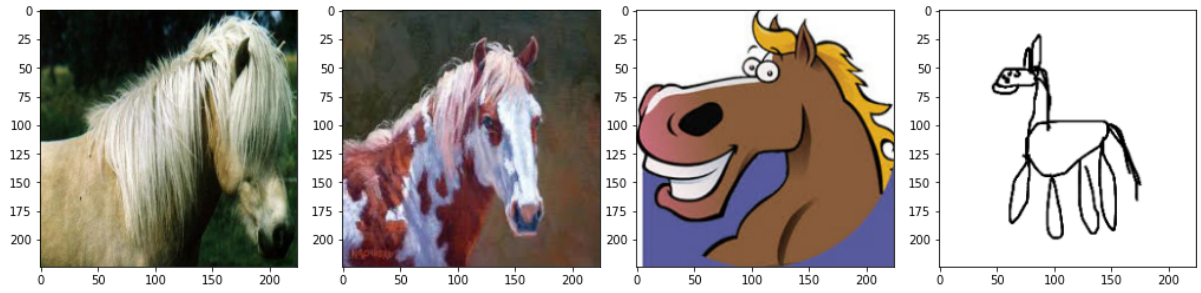
Roberto Franceschi (s276243)

1 Introduction

In this report, we will explore an image classification problem using different domains. In all applications of CNNs experimented during this course, we assumed that our training data representative of the underlying distribution. However, if the inputs at test time differ significantly from the training data, the model might not perform very well. For this homework, we have no access at training time to the distribution (i.e. domain) where the test images are taken from. In order to tackle the issue, we will use a modified version of the AlexNet [1] deep neural network structure that allows not only to classify input images in the source domain but also to transfer this capability to the target domain. Such task is called *domain adaptation (DA)* in deep learning research and has several applications in real-life situations.

For this analysis we will use the *PACS dataset* [2], which contains overall 9991 images, split unevenly between 7 classes and 4 domains: Art painting, Cartoon, Photo, Sketch. Figure 1 shows as an example some images of class 'horse' taken from this dataset with the correspondent domain label. As can be clearly seen from the example there are lots of differences between domains.

Our goal will be to train the network on images from the Photo domain and then to be able to correctly classify samples from the Art domain. Furthermore, *Cross Domain Validation* will be taken into account in the attempt of finding the best hyperparameters to increase the overall performance of the net.



(a) *(b)* *(c)* *(d)*

Figure 1: *Sample images from the PACS dataset one for each domain: (a) photo, (b) art painting, (c) cartoon, (d) sketch*

Additionally, in *Table 1* and *Figure 2* are summarized the distributions of images per domain and classes. The original size of each image is 227x227 pixels (with 3 channels, i.e. RGB) and their distribution across the 7 classes has the following parameters: mean of about 1427 images per class and a standard deviation of 269.4 images per class.
It is also interesting to note that the Photo dataset, the one we will use for training the neural network, is the smallest one, only 1670 samples (wrt a mean of 2497,7 images per domain).

	Dog	*Elephant*	*Giraffe*	*Guitar*	*Horse*	*House*	*Person*	***Total domain***
Photo	189	202	182	186	199	280	432	***1670***
Art Painting	379	255	285	184	201	295	449	***2048***
Cartoon	389	255	285	184	201	295	449	***2344***
Sketch	772	740	753	608	816	80	160	***3929***
Total class	***1729***	***1654***	***1566***	***1113***	***1540***	***943***	***1446***	

Table 1: *Number of images in each class and each domain*

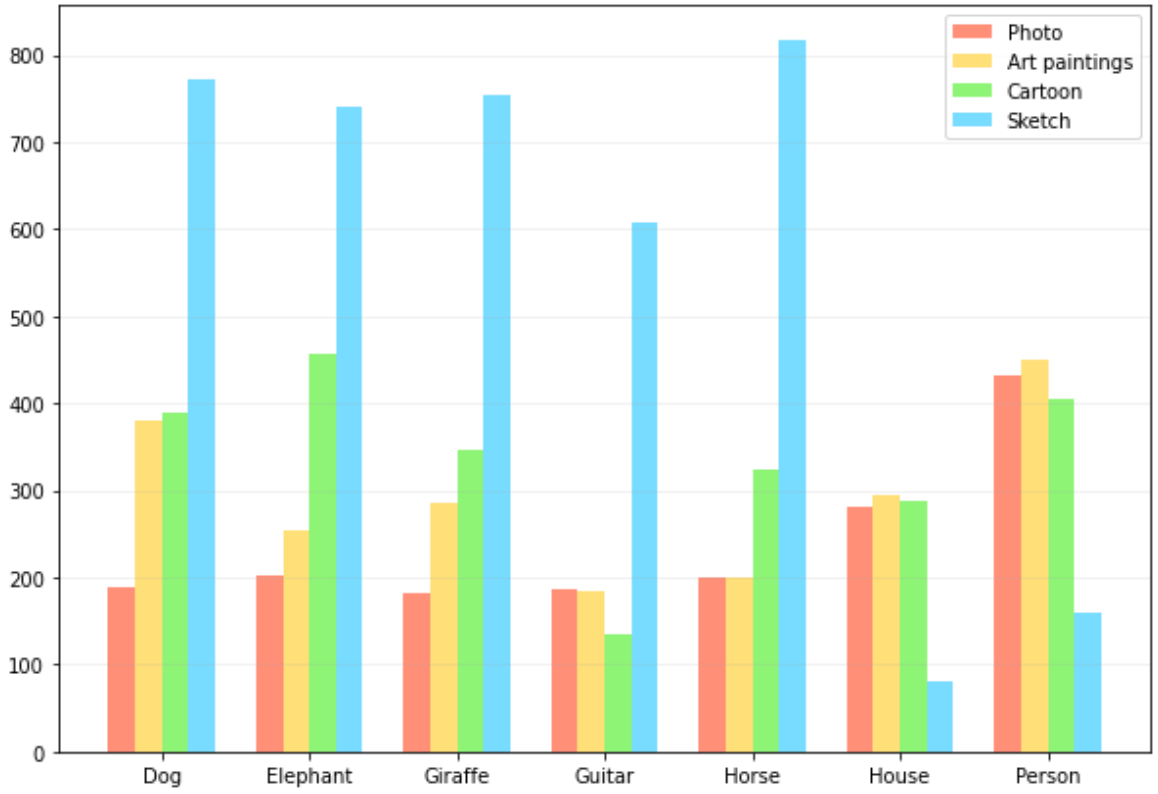


Figure 2: *Distribution of number of images in each class and each domain*

2 Network setup

The basic idea behind the DANN implementation presented in [3] is to learn a model that can generalize well from one domain to another, while ensuring that the internal representation of the neural network contains no discriminative information about the origin of the input (source or target). As already mentioned above, to implement this idea we will use a modified version of the original AlexNet. In particular, we have two parallel branches of fully connected layers after the convolutional section. The first one is the one from the original net with an output of 7-dimensional vector, corresponding to the classes in the PACS dataset.
Instead, the parallel branch aims to determine whether an image comes from the source or the target domain and it is connected to the feature extractor via a gradient reversal layer that multiplies the gradient by a certain negative constant during the backpropagation-based training. So, it will have as output a 2d vector, since it plays the role of a binary classifier.
However, during training we will try to confuse the network so that it is not able to tell the difference between the domains: by doing so during classification we can indifferently provide images from any domain to the network and expect that they are classified correctly.
The following figure shows what is described above, in particular includes a feature extractor (blue) of the original AlexNet, a label classifier (blue) and a domain classifier (pink).

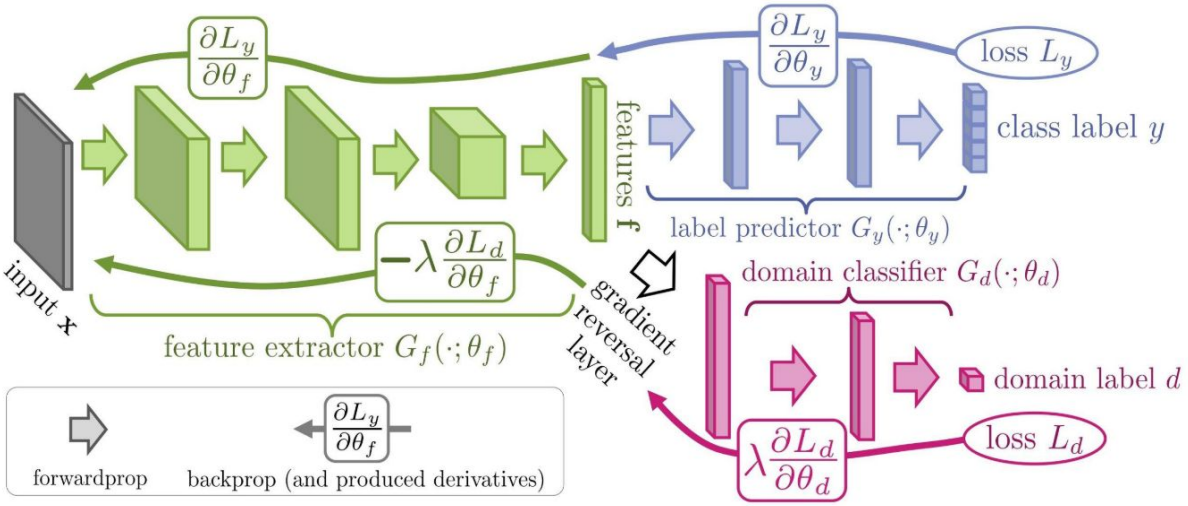


Figure 3: *Domain-adversarial neural network architecture by Ganin et al. [3]*

Before feeding the network, the desired transformation has been computed on the input images, normalizing them, and adapting their size to the input required by AlexNet, which is 224x224 pixels (using a `transforms.CenterCrop`).

The PACS dataset and code, together with results, can be found at: `https://github.com/robertofranceschi/DANN`.

3 Implementation of DANN adaptation

To implement what is described in Section 2 we define a new class `DANN_AlexNet` which implements the network, just like the class `AlexNet` in PyTorch. This class define the blocks

of the network: the first will contain the convolutional layers (`self.features`), while the other two will have the two parallel fully-connected layer (`self.classifier` for the label classifier and `self.GD` for the discriminator). Afterwards we have to change the logic of the forward function in order to allow to take the parallel branch when needed (this is done introducing the parameter `alpha`). When `alpha` is `None` we assume that we are training with supervision, instead if we pass alpha, we are training the discriminator. Likewise, the backward function has to propagate the gradient from the parallel branch back to the convolutional layers.
Furthermore, both the original network classifier and the new domain classifier has been initialized with the weights of the network as in the version of the AlexNet pretrained on the ImageNet dataset. In order to do this it is sufficient to load the parameters by means of the function `load state dict` provided in the original implementation of pytorch.

4 Domain adaptation using art painting as target

At the beginning, we train a traditional AlexNet and see how it performs in case of a domain adaptation problem. In this first part, runs were carried out without considering the *cartoon* and *sketch* domains as validation sets. Therefore, the network was trained on the *photo domain* and tested on *art painting* without domain adaptation (point 3A in the assignment) and consequently with DANN adaptation (point 3B). The results regarding this part are shown in the following table.

	Test accuracy without DANN	*Test accuracy with DANN*	*% Increment*
LR = 5e-3 BS = 256	51.35 % (± 1.0%)	54.52 % (± 0.1%)	+ 3,17%
LR = 5e-3 BS = 128	51.90 % (± 1.7%)	52.46 % (± 2.8%)	+ 0,56%
LR = 1e-3 BS = 256	49.33 % (± 0.5%)	51.10 % (± 0.8%)	+ 1.77%
LR = 1e-3 BS = 128	50.03 % (± 0.3%)	51.60 % (± 1.8%)	+ 1.57%
LR = 1e-2 BS = 256	50.19 % (± 2.4%)	50.20 % (± 3.2%)	+ 0.01 %
LR = 1e-2 BS = 128	51.18 % (± 1.7%)	53.59 % (± 2.1%)	+ 2.40%

Table 2: *Comparison of different hyperparameter sets without adaptation and with DANN adaptation. The last column report the increment between the two presented strategies.*

Clearly, we see from the last column, as we expected, that we have a significant increment performing training with DANN adaptation.
Note that in the second case we did something equivalent to "cheating" in the field of machine learning because informations regarding the test domain (i.e. art painting) were used during training of the neural net (in the discriminator branch). As we will see in the next section different strategies has been implemented that make use of informations taken from the other domains (*cartoon* and *sketch*).

5 Cross Domain Validation without adaptation

In this second step of our analysis we will consider also the others domains as validation sets. So, we will train the network on images coming from the source domain (i.e. Photo) and then we will test it on the target domain (i.e. art painting), without giving any informations to the network about the difference between the domains (i.e., without DANN adaptation).
The hyperparameters we are going to tune in this first part are learning rate and batch size. The other hyperparameters are fixed and have the following values:

- Epochs: 30
- Step size: 20
- Weight decay: $5 \cdot 10^{-5}$
- Gamma: 0.1

They define respectively how many times we will pass through the entire dataset in the course of the training phase, the number of epochs after which the learning rate is decreased, the regularization parameter and the decreasing factor of the learning rate. The optimizer used in our analysis is the SGD with momentum.
For each of the two validation domains we perform a grid search across different values of the hyperparameters *learning rate (LR)* and *batch size (BS)*. The results of the validation phase are shown in table 3 for the cartoon and sketch domain. In order to properly select the best hyperparameter set, we pick the results obtained on the two domains and choose the set that performed at best on average (see Table 4).

	BS = 256	*BS = 128*
LR = 5e-3	27.56 %	26.78 %
LR = 1e-3	24.59 %	26.02 %
LR = 1e-2	28.65 %	27.54 %

(a)

	BS = 256	*BS = 128*
LR = 5e-3	29.34 %	30.86 %
LR = 1e-3	30.27 %	29.91 %
LR = 1e-2	27.54 %	33.26 %

(b)

Table 3: *Comparison of different hyperparameter sets on the (a) cartoon and (b) sketch domains.*

	BS = 256	*BS = 128*
LR = 5e-3	28.45 %	28.82 %
LR = 1e-3	27.43 %	26.42 %
LR = 1e-2	29.28 %	**30.40 %**

Table 4: *Average of the validation results between the cartoon and sketch domain.*

The best score is obtained with LR=1e-2 and batch size=128. Finally, we test the network on the *Art painting* domain, which delivers an accuracy of 51.06 % (± 1.51 %).

6 Cross Domain Validation with domain adaptation

After implementing the modified neural network for domain adaptation, we are able to execute the task presented in the beginning of the report. In the training phase, our task is to optimize the parameters both of the label classifier and of the domain classifier. In order to train the discriminator, we use the images from the *cartoon* and *sketch* domains as target. Each training epoch consists of three steps as described in paper [3]:

1. training the classifier using images from the source domain,
2. training the discriminator using images from the source domain,
3. training the discriminator using images from the target domain.

For this analysis we selected as hyperparameters to tune the same as before (LR and BS) and the parameter alpha, which is a weighting parameter to control the gradient propagation from the parallel branch. Therefore, we perform a grid search between the parameters of Section 4 and different values of alpha, that are 0.01, 0.1, 0.25, 0.5 and 0.8. As before, we validate the network on the domains cartoon and sketch in order to find the best hyperparameter set. The average results of cartoon and sketch are reported in table 5.

	Alpha = 0.01		*Alpha = 0.1*		*Alpha = 0.25*	
	BS = 256	*BS = 128*	*BS = 256*	*BS = 128*	*BS = 256*	*BS = 128*
LR = 1e-4	21.54 %	23.13 %	22.43 %	24.32 %	26.98 %	30.28 %
LR = 5e-3	26.99 %	26.63 %	28.39 %	**35.95 %**	26.96 %	33.93 %
LR = 1e-3	25.34 %	28.14 %	26.97 %	26.45 %	17.03 %	18.40 %
LR = 1e-2	28.47 %	26.51 %	25.32 %	27.02 %	17.03 %	17.03 %

Table 5: *Average of the validation results between cartoon and sketch domains.*

From the results we can notice that as alpha increases it is necessary to lower the learning rate to make the neural network learn without there being divergence in one of the losses. This can be seen with alpha=0.25 and high learning rates, since the average accuracy value is below 20%.

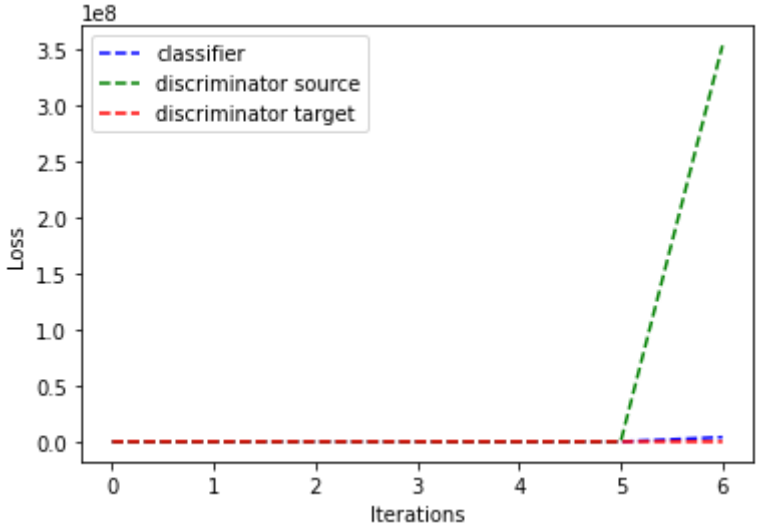


Figure 4: *Example of hyperparameters that lead the network to perform bad on validation*

Finally, the best score is obtained with LR=5e-3, alpha=0.1 and batch size=128. Finally the test accuracy (i.e. on Art Painting domain) found with the best set of hyperparameters is 52.31 % (± 1.72 %).

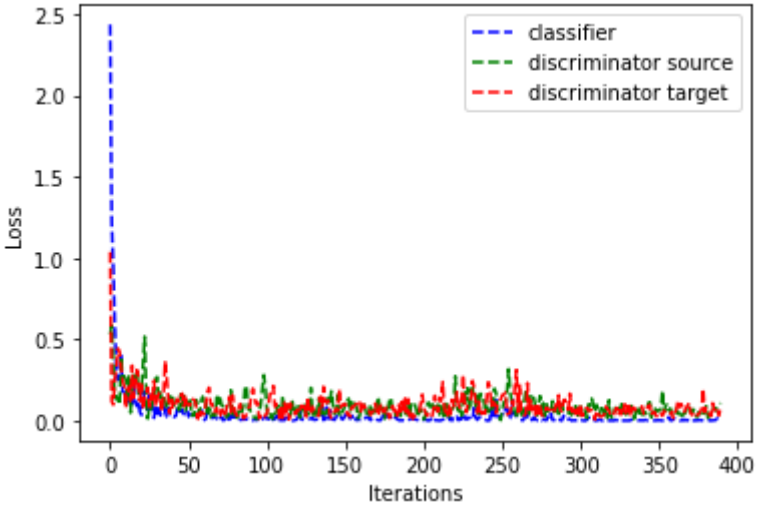


Figure 5: *Plot of the losses obtained with the best hyperparameters set*

Note that for the simplicity of this report some results with high alpha values are omitted (e.g. alpha 0.5 and 0.8), since experimentally has been seen that the more alpha grew the

worse the results are (because the discrimination loss tends to diverge after few epochs with high learning rates). For this reason further investigation has been done to reduce the number of epochs and consequently the step size (e.g. NUM EPOCHS = 10, STEP SIZE=6). Despite this effort, the model that performed better was the one proposed previously.

References

[1] Krizhevsky, Alex & Sutskever, Ilya & Hinton, Geoffrey. (2012). ImageNet Classification with Deep Convolutional Neural Networks. Neural Information Processing Systems. 25. 10.1145/3065386.

[2] Li, D., Yang, Y., Song, Y.Z., & Hospedales, T. (2017). Deeper, Broader and Artier Domain Generalization. In International Conference on Computer Vision.

[3] Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., & Lempitsky, V.S. (2016). Domain-Adversarial Training of Neural Networks. ArXiv, abs/1505.07818.