POLITECNICO DI TORINO

MSc in Data Science and Engineering

Machine learning and Deep learning

Homework 3

Professor:
Barbara Caputo

Student:
Paola Privitera

Academic Year 2019-2020

Index

Introduction

The aim of the project is to implement a DANN (Domain Adversarial Neural Network) for image recognition on the PACS dataset.

The idea beneath the domain adaptation concept is to align training features with test features (reducing the domain shift). To prove the point, one of the main struggles in Artificial Intelligence and Deep Learning is that not always is possible to test model results on dataset which are drawn from the same distribution of the training set. In fact, the ideal condition is when training and test data satisfy a set of goodness conditions such that they can be considered belonging to the same data distribution, needless to say that under this framework it is possible to achieve high quality results.

However, in real practice, training and test objects belong to different distribution which inevitably implies that the network is biased and therefore produces inaccurate predictions.

In order to highlight the implication of the above-mentioned problem, we created a DANN model using an augmented version of the AlexNet; indeed, the network is composed by a feature extractor and a label predictor, plus an additional gradient reversal layer – employed during the backpropagation training, to smooth the label distribution across the two domains used for training and test.

We analyzed the performances under two different scenarios: the first one called "without adaptation" which consists in training a classical AlexNet, opposed to the second case ("with adaptation") where the architecture used is the DANN.

Additionally, for each scenario we detailed two opposite approaches to address the validation phase issue. By estimating the predictions label directly from the test set, disregarding any form of a-priori validation controls, it might not provide consistent outputs. Therefore, it is important to understand the impact of validation sets for hyperparameter tuning in the overall domain adaptation process. The first approach does not involve any validation, whilst the second includes an independent validation set (later indicated as Cross Domain Validation) to derive the best hyperparameters configuration to deploy during test phase.

Finally, we summarized and discussed results in dedicated sections.

1 Dataset

The PACS dataset consists in 9,991 images divided in 4 domains: *Photo*, *Art painting*¸ *Cartoon* and *Sketch.* Each picture represents an object belonging to one of the following categories: dog, elephant, giraffe, guitar, horse, house and person.

By using the ImageFolder class, we treated each domain as a PyTorch dataset. As the usual data science practice requires, we attributed to each domain the following functions:

- **Photo** domain, referred hereafter as *source domain*, represents the training set,
- **Art painting** domain, also indicated as *target domain*, represents the test set,
- and **Cartoon** and **Sketch** domains, as two distinct validation sets.

To adapt the image resolution to input size of AlexNet architecture, we centrally cropped all images to a specific dimension (224x224). Since the model is pretrained on the ImageNet dataset, we performed a normalization through its mean and standard deviation to adapt the images to the network dynamics.

2 Implementation of the model

The initial implementations of the DANN applied to small networks for digits datasets. In the present work, the aim is to recreate the same network architecture using an enriched version of the AlexNet.

The proposed architecture includes a deep *feature extractor* (composed exclusively by convolutional layers) and a *deep label predictor* (main classifier), which together form a standard feed-forward architecture (classical AlexNet). Unsupervised domain adaptation is achieved by adding a *domain classifier* connected to the feature extractor via a *gradient reversal layer* that multiplies the gradient by a certain constant with negative sign (-α, initially set by default and optimized as an hyperparameter later on) during the backpropagation-based training. Otherwise, the training proceeds standardly and minimizes the label prediction loss (for *source* examples) and the domain classification loss (for all samples). Gradient reversal ensures that the feature distributions over the two domains are made similar (as indistinguishable as possible for the domain classifier), thus resulting in the domain-invariant features.

Since the main classifier’s goal is to predict the image categories intra-domain, it is trained only with data from the source domain. On the other hand, the domain classifier is trained with both source and target data as its goal is to discriminate whether an image belongs to the source domain or the target domain. In doing so, the domain classifier tries to minimize the corresponding prediction error.

The final component of the architecture is the gradient reversal layer. Its function is inverting the gradients of the domain classifier to the feature extractor (during the backpropagation). The features of our backbone are trained such that the prediction error of the domain classifier is maximized. On the contrary, the features are normally trained basing on the classification loss.

From the implementation standpoint, we added to a classical AlexNet a separate branch for the domain classifier with the same architecture of AlexNet fully connected layers. In plain words, we added a new densely connected branch with two output neurons, telling whether an image belongs to the source or the target domain. Both the original network and the new classifier have been initialized with the same weights of ImageNet (pretrained network). In addition, we modified the forward function, by creating a flag which indicates whether a data batch should be attributed either to the main classifier or the domain classifier.

To estimate the difference between the prediction of the labels from the ground truth, we used the Cross Entropy Loss which is defined as the negative logarithm of the softmax function; furthermore, we used the Stochastic Gradient Descent (SGD) with momentum as optimization technique to update the weights based on loss.
Finally, we defined the scheduler which is responsible for the decay of the learning rate after a given number of epochs.

As one might have noticed, we did not mention the usage of validation sets in the network architecture explained so far. The validation phase is an ongoing concern in domain adaptation, indeed, estimating the predictions directly from the test set represents a forced and naïve procedure, which might not provide with consistent and trustworthy results. Therefore, it is fair to analyze the impact of validation sets for hyperparameter tuning in the overall domain adaptation process. In the following paragraphs, we detailed two opposite approaches: the first one which does not involve any validation, and the second which includes the *Cartoon* and *Sketch* domains as validation sets to derive the best values for the hyperparameters. Nonetheless, both approaches have been carried out initially by deploying the classical AlexNet (i.e. without adaptation) and then using the DANN (i.e. with adaptation).

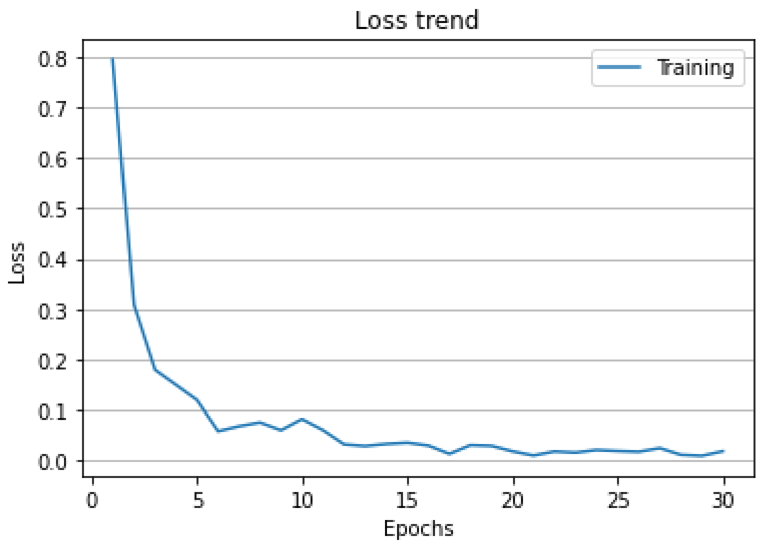
3 Train and test without validation

As mentioned, the first approach does not require any validation set. The initial set of hyperparameters is the following:

BATCH SIZE	= 256
LR	= 0.001
MOMENTUM	= 0.9
WEIGHT DECAY	= 0.00005
EPOCHS	= 30
STEP SIZE	= 20
γ	= 0.1
α	= 0.01

3.1 Without adaptation

The training is performed using the *Photo* dataset and later the model is tested on the *Art painting* dataset without the domain adaptation, which means the second branch (domain classifier) is not utilized at this stage of the process.



The graph above shows the loss values on training set for each epoch. The corresponding final accuracy stood at approximately 49.2%.

3.2 With adaptation

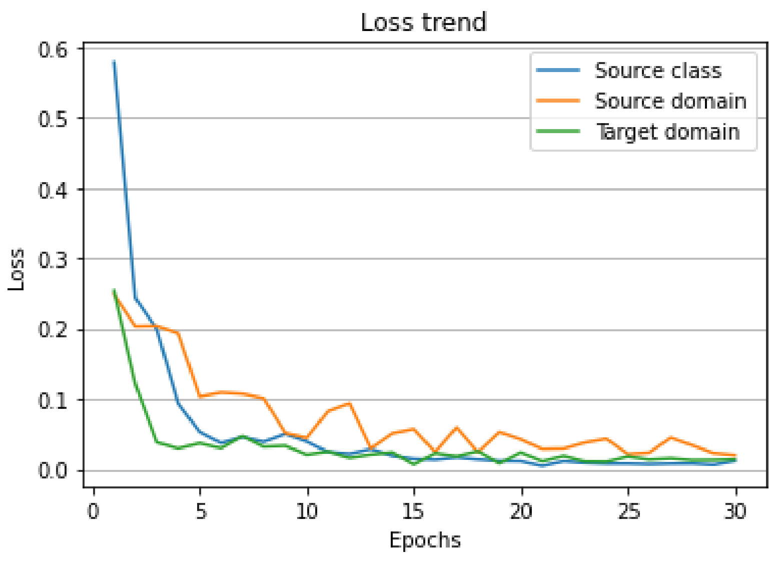
The introduction of the adaption implies that, the network is jointly trained on the *Photo* and *Art painting* datasets (the latter without labels) and tested solely on the *Art painting* dataset.

Each training iteration is divided into three steps:

a. train the main classifier using the *source* domain,
b. train the discriminator on the *source* domain,
c. train the discriminator on the *target* domain.

At the end of each iteration, the network weights are adjusted according to the values of the resulting gradients.

It should be pointed out that due to the different dimension of the *source* and *target* domain, the number of iterations on each individual set would be different. Supposing the *target* domain has length k and the *source* domain has length s, with $s < k$, we iterated on the source domain an additional (k-s) times over again the (k-s) first data points on the *source* domain, so that none of the elements of the *target* domain is missed.



The graph above shows the loss value for each of the steps of the training at each epoch, with a final accuracy on the test of approximately 50.9%.

4 Cross Domain Validation

As previously mentioned, the validation phase relies onto two datasets: *Cartoon* and *Sketch*. In particular, we performed a combined grid search and the corresponding performances on *Photo* to *Cartoon* and on *Photo* to *Sketch* have been measured and the resulting accuracies have been averaged for each hyperparameters' set. Finally, we trained the model on source domain and tested on target domain, using as best values, the configuration of hyperparameters that showed the highest accuracy.

4.1 Without adaptation

The hyperparameters used for the grid search (accounting for each possible combination of these values), keeping all the other parameters by default, are the following:

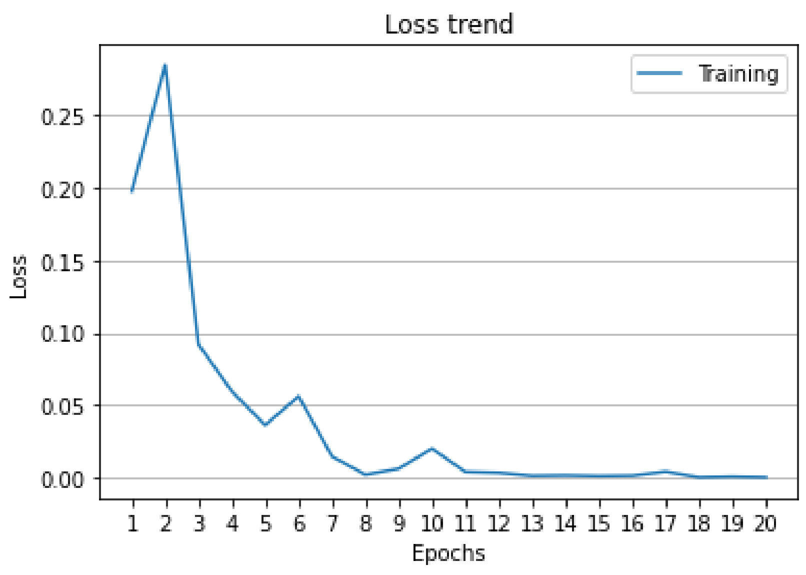
LR = [0.001, 0.01]

NUM_EPOCHS = [20, 30]

STEP_SIZE = [10, 20]

The best validation accuracy is 33.16% corresponding to the following combination of hyperparameters: learning rate = 0.01, number of epochs = 20 and step size = 20.

The chart below shows the loss curve dynamics on the training set using the best hyperparameters found.



The resulting accuracy on the test set is 50.97%, an increase of approximately 2 percentage points compared to 49.20% of the model without validation.

4.2 With adaptation

In this case, we implemented two different networks, the first one using *Cartoon* as validation set and the second one using *Sketch*. We trained the two models using each combination of hyperparameters and averaged the corresponding accuracies. Hence, we carried out the same procedure for testing as in the previous case.

Since the execution was computationally expensive, we decided to reduce the number of epochs to 10. The hyperparameters used for tuning are:

LR = [0.001, 0.01]

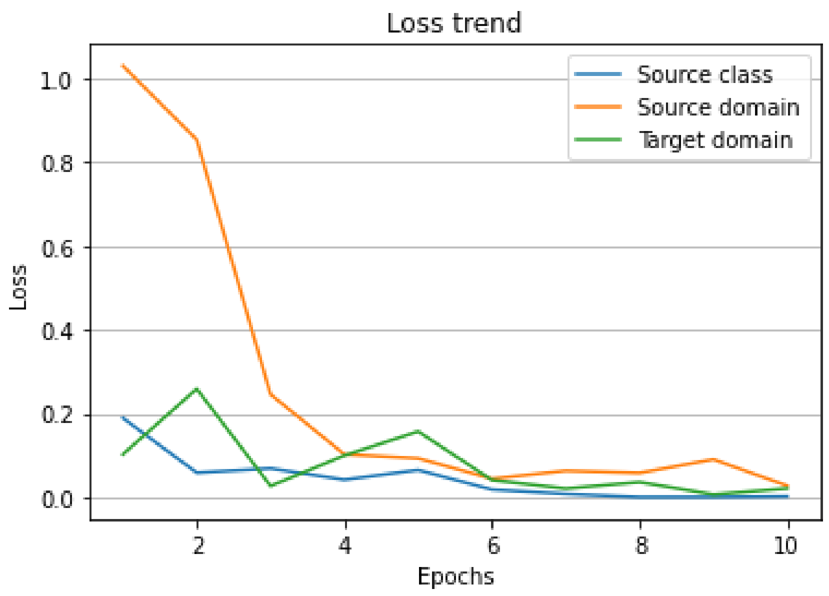
ALPHA = [0.01, 0.03, 0.5]

The following table shows the corresponding average accuracies:

LR / ALPHA	**0.01**	**0.03**	**0.5**
0.001	23.14%	23.48%	diverges
0.01	34.89%	**35.82%**	diverges

The highest value of 35.82% corresponds to the combination LR-Alpha (0.01, 0.03).

The next chart shows the loss curve dynamics on the training set using the best hyperparameters found.



The resulting accuracy on the test set is 52.88% . Analogously, in this case we noticed an increase of approximately 2 percentage points compared to 50.90% of the model trained on DANN without validation.