Machine Learning and Deep Learning
01TXFSM
Politecnico di Torino

Homework 3

Student: Riccardo Mereu Student ID: s277004

1 Introduction

The goal of this homework is to implement Domain Adversarial Neural Network (DANN), a Domain Adaptation algorithm, and apply it on the PACS dataset using as a backbone AlexNet.

1.1 PACS Dataset

The PACS dataset is a Domain Generalization benchmark dataset introduced by [2]. This dataset covers four domains (Photo, Sketch, Cartoon, Art Painting), and seven common categories 'dog', 'elephant', 'giraffe', 'guitar', 'horse', 'house', 'person'. The total number of images is 9991, divided into 1670 images for the Photo domain, 2048 for Art Painting, 2344 for Cartoon and 3929 for Sketch. The class distributions on PACS dataset and the number of images in each domain are reported in Figure 1.

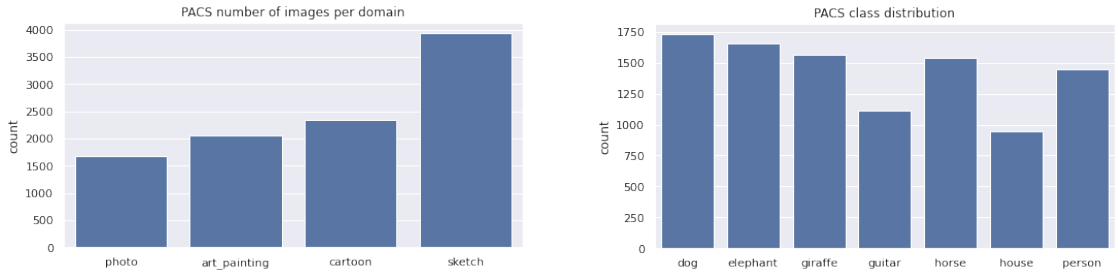

Figure 1: Class distribution of the different domains of the PACS dataset

The class distribution across the different domains are reported in Figure 2.As we can see from the plots, the classes are highly imbalanced and the number of images for each domain is limited.

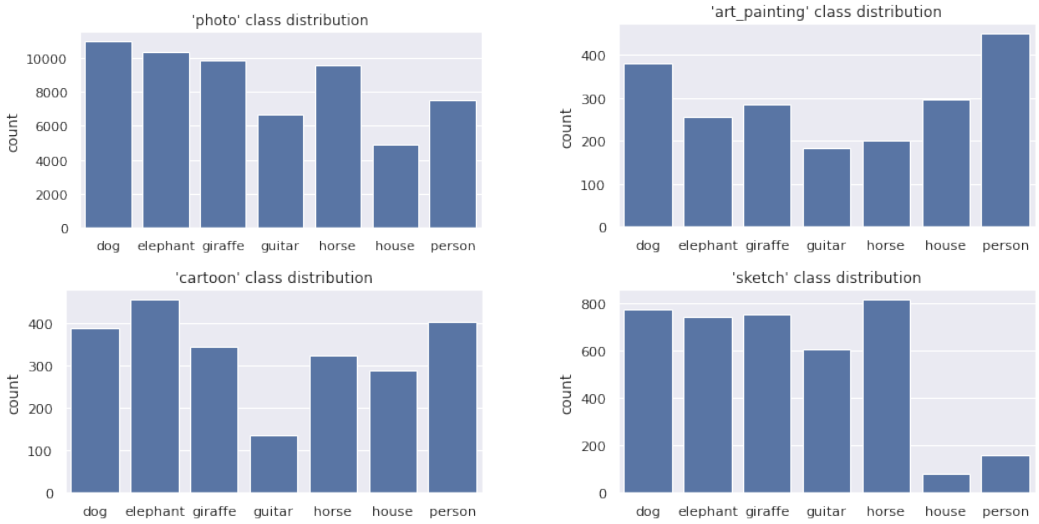

Figure 2: Classi distribution of the different domains of the PACS dataset

2 Domain Adaptation

Over the past few years, machine learning has achieved great success and has benefited real-world applications. However, collecting and annotating datasets for every new task and domain are extremely expensive and time-consuming processes and sufficient training data may not always be available.

Unfortunately, due to many factors (e.g., illumination, pose, and image quality), there is always a distribution change or domain shift between two domains that can degrade the performance.
Domain Adaptation (DA) is a particular case of Transfer Learning (TL) that utilizes labelled data in one or more relevant source domains to execute new tasks in a target domain. The following definition of DA, taken from [3], here refers only to DA, while in that paper covers the broader definition of transductive transfer learning.
Given a source domain $\mathcal{D}_S$ and a corresponding learning task $\mathcal{T}_S$, a target domain $\mathcal{D}_T$ and a corresponding learning task $\mathcal{T}_T$, domain adaptation aims to improve the learning of the target predictive function $f_T(\cdot)$ in $\mathcal{D}_T$ using the knowledge in $\mathcal{D}_S$ and $\mathcal{T}_S$, where $\mathcal{D}_S \neq \mathcal{D}_T$ and $\mathcal{T}_S \neq \mathcal{T}_T$.

Various shallow DA methods have been proposed to solve a domain shift between the source and target domains. In this homework, the focus will be on a Deep DA method based on Adversarial Domain Adaptation.

2.1 DANN: Domain Adversarial Neural Network

Domain Adversarial training of Neural Network is a training method for DA in the context of feedforward neural network architectures proposed by Ganin et. al [1]. With this approach, the networks are trained on labeled data from the source domain and unlabeled data from the target domain (no labeled target-domain data is necessary).
The method is built over the idea that to achieve an effective domain transfer the predictions of the model must be based on features that cannot discriminate between the source domain and the target domain. In other words, these features must be domain agnostic.

The networks trained with DANN learn a feature mapping that is both discriminative for the main learning task on the source domain $\mathcal{T}_S$ and invariant with respect to the shift between the domains. This goal is achieved by jointly optimizing two classifiers: a label predictor that predicts class labels ($\mathcal{G}_y(\cdot\,;\theta_y)$), used both during training and test time; a domain classifier that discriminates between the source and the target domains during training ($\mathcal{G}_d(\cdot\,;\theta_d)$).
The parameters of the classifiers, $\mathcal{G}_y$ and $\mathcal{G}_d$, are optimized to minimize their error on the training set, composed of labeled data from the source domain and unlabeled data from the target domain. The parameters of the underlying feature mapping, $\mathcal{G}_f(\cdot\,;\theta_f)$, are optimized to minimize the loss of the label classifier and maximize the error of the loss of the domain discriminator. This adversarial training of the feature mapping is achieved using a gradient reversal layer that leaves the input unchanged during forward propagation and reverses the gradient by multiplying it by a negative scalar during the backpropagation.

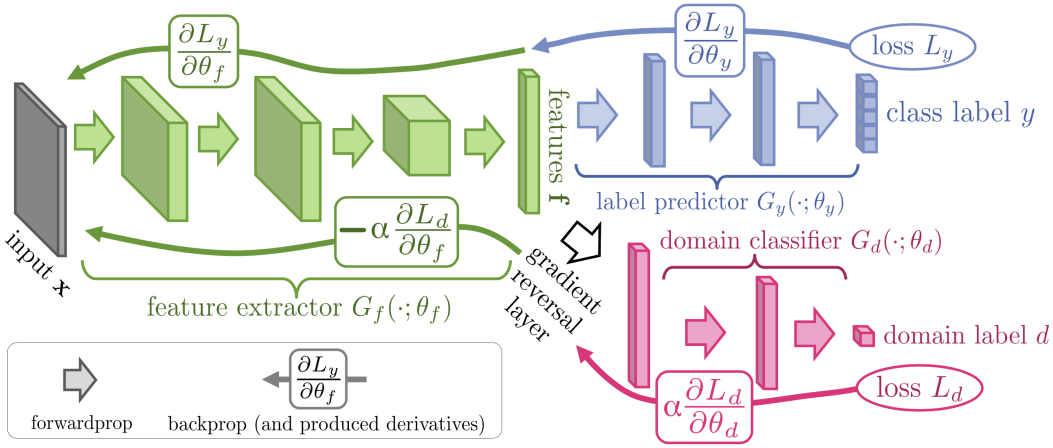

Figure 3: DANN architecture for deep feed-forward neural networks.

The DANN training can be generalised to modern deep convolutional neural networks. In this homework, the network under analysis is AlexNet, that will be trained for the DA task with and without DANN. Figure 3 contains a graphical representation of the architecture needed for DANN training. The feature extractor $\mathcal{G}_f(\cdot\,;\theta_f)$ (green) and the label predictor $\mathcal{G}_y(\cdot\,;\theta_y)$ (blue) together form the standard feed-forward architecture. In this specific case, AlexNet's convolutional layers are the feature extractor, while the label predictor is composed of the fully connected layers on top of them. The features obtained by $\mathcal{G}_f$ are sent both to $\mathcal{G}_y$ but also to the domain classifier $\mathcal{G}_d(\cdot\,;\theta_d)$ (purple). The latter branch has the same architecture of the label classifier, but with only two output neurons. This branch is connected to the feature extractor via a gradient reversal layer.

3 Experiments and results

This section contains a description of the different experiments conducted in this homework. The first part covers point 3 of the assignment which consists of a hyperparameter optimization for the

baseline, i.e. an AlexNet model finetuned on the source domain dataset, and AlexNet trained with DANN approach. Both models are trained on Photo and tested on Art Painting ($P \rightarrow A$).
In the second part, a different methodology was used to validate the hyperparameters. The other two domains of PACS dataset, Cartoon (C) and Sketch (S), are used as target domains to choose the best hyperparameters. The hyperparameters that lead to the best average accuracy on $P \rightarrow C$ and $P \rightarrow S$, are then used to train again the model for the DA from Photo to Art Painting.
The last section covers the use of an exponential policy for the hyperparameter α. This technique is the same used in the original paper [1].
For the training of the models, dataset augmentation is used on the training set. The transformations applied to the input images are RandomHorizontalFlip and ColorJitter.

3.1 Point 3: without validation

3.1.1 AlexNet

The first section deals with point 3.A of the assignment. For this request, AlexNet is trained without DANN using the entire 'photo' dataset as training set and the entire 'art_painting' dataset as test set. The hyperparameter optimization is done with a grid search over learning rate and weight decay. They are chosen respectively from this two sets $LR = \{1 \cdot 10^{-2}, 5 \cdot 10^{-3}, 1 \cdot 10^{-3}\}$ and $WD = \{5 \cdot 10^{-4}, 5 \cdot 10^{-5}\}$. For each run, the hyperparameters kept fixed are: the number of epochs equal to 30 epochs, the SGD momentum equal to 0.9 and the batch size equal to 256 for both sets.

Learning rate	**Weight Decay**	**Source Loss**	**Target Loss**	$P \rightarrow A$ **Accuracy**
$1 \cdot 10^{-2}$	$5 \cdot 10^{-4}$	0.0052	3.7614	0.5400
	$5 \cdot 10^{-5}$	0.1791	2.5901	0.5396
$5 \cdot 10^{-3}$	$5 \cdot 10^{-4}$	0.0029	3.4650	0.5464
	$5 \cdot 10^{-5}$	0.0125	3.1597	0.5400
$1 \cdot 10^{-3}$	$5 \cdot 10^{-4}$	0.0238	2.5010	0.5015
	$5 \cdot 10^{-5}$	0.0204	2.3898	0.5156

Table 1: AlexNet PA

Table 1 contains the results of the grid search. The highlighted row contains the best model, which achieves an accuracy of 54.64% on the test set. This model will be used in this point as a baseline to compare the results obtained with the DANN training. For each hyperparameter combination, the table contains also the values of the loss achieved on the source dataset and the loss achieved on the target dataset.

3.1.2 DANN

In this section, the results obtained through a grid search hyperparameter optimization on the models trained with DANN training using in the training phase the entire 'photo' dataset with its labels plus the unlabeled data of the target dataset 'art_painting'. The grid search is done over the same LR and WD sets with the addition of different values of α hyperparameter taken from the set $A = \{0.01, 0.05, 0.1, 0.15, 0.2\}$.

α	**Learning rate**	**Weight Decay**	**DANN Loss**	**Target Loss**	$P \rightarrow A$ **Accuracy**
0.2	$5 \cdot 10^{-3}$	$5 \cdot 10^{-4}$	0.6496	2.6185	0.5605
0.1	$5 \cdot 10^{-3}$	$5 \cdot 10^{-5}$	0.1094	3.8737	0.5601
0.05	$5 \cdot 10^{-3}$	$5 \cdot 10^{-4}$	0.0322	3.4781	0.5566
0.01	$5 \cdot 10^{-3}$	$5 \cdot 10^{-4}$	0.0404	3.7653	0.5562
0.01	$1 \cdot 10^{-2}$	$5 \cdot 10^{-4}$	0.0475	2.9170	0.5532

Table 2: AlexNet trained with DANN for $P \rightarrow A$ domain adaptation

Table 2 contains the top five results obtained with the grid search described above. The results are sorted in descending order of accuracy over the Art Painting dataset. The model that achieved the best accuracy of 56.05% on the test dataset was trained using as hyperparameter $\alpha = 0.2$, learning rate equal to $lr = 5 \cdot 10^{-3}$ and weight decay equal to $wd = 5 \cdot 10^{-4}$.
Table 2 contains only a part of the results obtained, the complete output results of the grid search are reported in table 10 in the Appendix. Note that the set A contains values of α up to 0.2. The reason behind this choice is related to the fact that for higher α values the model diverges.
The best result obtained in this section is better than the one obtained with the AlexNet baseline of Section 3.1.1 but using this validation methodology is not completely fair. In fact, the choice of the models for both the baseline and the DANN trained model is based on the accuracy computed on the test set, which theoretically should not be available at training time. The next section will focus on a possible approach to overcome this flaw.

3.2 Point 4: validation over Cartoon and Sketch dataset

A correct way to perform validation for Domain Adaptation is still an open problem. As stated before a correct validation step should not look at the results in the test set. A possible solution proposed in the assignment is to exploit the other two domain datasets, Cartoon and Sketch.
The idea is to perform the validation by evaluating the hyperparameters for our task $P \rightarrow A$, by running a grid search on Photo to Sketch ($P \rightarrow S$) and Photo to Cartoon ($P \rightarrow C$).

3.2.1 AlexNet

As in section 3.1.1, the hyperparameters tuned are the learning rate and the weight decay. They are chosen from sets *LR* and *WD* defined before. The other hyperparameters, i.e. SGD momentum and batch size, are kept fixed. Since in the following phase, the target set Art Painting is not available at training time, it is important to choose the right value for the number of epochs. For this reason, the hyperparameters will be chosen based on the best average accuracies. The accuracies are evaluated over the validation dataset at each epoch. Keeping track of the epochs in which the best accuracies are obtained is useful to tune the proper number of epochs to use when training the model for Photo to Art Painting.

Table 3 contains the results obtained in the grid search. The highest average accuracy on the test sets is 35.16% obtained with a learning rate of $1 \cdot 10^{-2}$, weight decay equal to $5 \cdot 10^{-5}$. The model obtained an accuracy of 33.34% on $P \rightarrow C$ at the 8^{th} epoch and 36.99% on $P \rightarrow S$ again in the 8^{th} epoch. These hyperparameters were used to train again AlexNet for 8 epochs on the entire Photo dataset.

		$P \to S$				$P \to C$				
LR	**WD**	**Source Loss**	**Target Loss**	**Target Acc.**	**Ep.**	**Source Loss**	**Target Loss**	**Target Acc.**	**Ep**	**Avg Acc.**
$1 \cdot 10^{-2}$	$5 \cdot 10^{-4}$	0.0087	2.6910	0.3258	10	0.0131	4.9472	0.3178	14	0.3218
	$5 \cdot 10^{-5}$	0.0212	3.3979	0.3334	8	0.0235	3.3950	0.3699	8	0.3516
$5 \cdot 10^{-3}$	$5 \cdot 10^{-4}$	0.0045	3.4384	0.3161	22	0.0150	4.8078	0.2863	29	0.3012
	$5 \cdot 10^{-5}$	0.0155	2.6920	0.3324	26	0.0027	5.2040	0.3451	11	0.3388
$1 \cdot 10^{-3}$	$5 \cdot 10^{-4}$	0.4696	1.8806	0.2721	26	0.0187	4.1475	0.2611	10	0.2666
	$5 \cdot 10^{-5}$	0.0235	2.7287	0.2219	19	1.2483	1.9831	0.2735	23	0.2477

Table 3: AlexNet for domain adaptation on $P \to C$ and $P \to S$

The AlexNet model trained using these hyperparameters achieves an accuracy on Art Painting dataset of 49.41%, which will be used as a baseline for the results obtained with this validation approach using DANN training. The results of this run are in Table 4.

LR	**WD**	**Epochs**	**Source Loss**	**Target Loss**	$P \to A$ **Acc.**
$1 \cdot 10^{-2}$	$5 \cdot 10^{-5}$	8	0.0357	2.7709	0.4941

Table 4: AlexNet on $P \to A$

3.2.2 DANN

This section deals with point 4.B and contains the results obtained validating the different hyperparameters for DANN training on Cartoon and Sketch dataset. As in the previous section, a grid search is performed over the values taken from the sets $LR = \{1 \cdot 10^{-2}, 5 \cdot 10^{-3}, 1 \cdot 10^{-3}\}$, $WD = \{5 \cdot 10^{-4}, 5 \cdot 10^{-5}\}$ and $A = \{0.01, 0.05, 0.1, 0.15, 0.2\}$.

			$P \to S$				$P \to C$					
α	**LR**	**WD**	**Source Loss**	**Target Loss**	**Target Acc.**	**Ep.**	**Source Loss**	**Target Loss**	**Target Acc.**	**Ep.**	**Avg Acc.**	**Ep.**
0.05	0.01	5E-04	2.78907	5.39774	0.41334	22	0.0859	5.3033	0.4735	22	0.4434	22
0.10	0.005	5E-04	0.06044	2.41813	0.34029	8	0.0998	6.5344	0.5158	30	0.4280	19
0.05	0.01	5E-05	0.14011	2.08930	0.38509	5	0.0761	5.1821	0.4275	14	0.4063	10
0.01	0.01	5E-05	0.01491	2.96167	0.34996	18	0.0657	3.6403	0.3810	10	0.3655	14
0.05	0.005	5E-05	0.01247	2.82493	0.31484	12	0.0135	5.9904	0.3938	30	0.3543	21

Table 5: DANN on $P \to C$ and $P \to S$

The five best results obtained with this hyperparameter search are reported in Table 5. Table 11 in the Appendix contains all the results obtained.

α	**LR**	**WD**	**Epochs**	**Source Loss**	**Target Loss**	$P \rightarrow A$ **Acc.**
0.05	$1 \cdot 10^{-2}$	$5 \cdot 10^{-4}$	22	0.082	3.7686	0.5376

Table 6: DANN for domain adaptation on $P \rightarrow A$

Inspecting the results in Table 5, we can see that the best average accuracy is 44.34% obtained using $\alpha = 0.5$, $lr = 0.01$ and $wd = 5 \cdot 10^{-5}$ in the 22^{nd} epoch for both $P \rightarrow C$ and $P \rightarrow S$ tasks. A new model using these hyperparameters is trained using DANN over the entire labeled Photo dataset and the unlabeled data of the Art Painting. This model achieves 53.76% of accuracy on the test set. This result is better than the one obtained with the same validation technique with AlexNet of the previous section but is lower if compared to the result obtained with DANN in Section 3.1.2. As explained before, with this validation approach the model at training time does not have access to the test data. In the previous section, the training was driven by the accuracy on the test set instead, on the contrary here the performance on the test data is correctly checked only at the end of the training.

3.3 Extra Points

In the DANN paper [1], the authors did not use a fixed value for both hyperparameters α and lr. The learning rate is adjusted during SGD using an ad hoc learning scheduler, following this formula

$$\eta_p = \frac{\eta_0}{(1 + \lambda \cdot p)^\beta}$$

where p is the training progress linearly changing from 0 to 1, $\lambda = 10$ and $\beta = 0.75$. η_0 is the initial learning rate that can be tuned with a grid search. The domain adaptation parameter α is initiated at 0 and gradually changed to 1 using the following schedule:

$$\alpha_p = \frac{2}{1 + \exp(-\gamma \cdot p)} - 1$$

However, to properly tune to this specific task the value of this hyperparameter the schedule is slightly modified to

$$\alpha_p = \left(\frac{2}{1 + \exp(-\gamma \cdot p)} - 1 \right) \cdot \alpha_*$$

then, α changes during the training process as a function of p from 0 to a generic value α_*.

In this section, two additional grid search were performed to identify the best values for α_* and the learning rate lr_0. The experiments were conducted using the second validation approach used in Section 3.2. The models are trained both using the learning rate schedule proposed in the paper or keeping the learning rate fixed.
The values for α_* are chosen from the set $A = \{0.01, 0.1, 0.2, 0.5, 0.7, 1\}$, while the one for the learning rate are from set $LR = \{1 \cdot 10^{-2}, 5 \cdot 10^{-3}, 1 \cdot 10^{-3}\}$. The other hyperparameters tuned are the weight decay taken from set $WD = \{5 \cdot 10^{-4}, 5 \cdot 10^{-5}\}$ and the number of epochs. The latter is chosen as the average between the epochs in which the model achieves the best test accuracy for Cartoon and Sketch datasets.

			$P \to S$				$P \to C$					
α_*	$\mathbf{lr}_0$	**wd**	**Source Loss**	**Target Loss**	**Target Acc.**	**Ep.**	**Source Loss**	**Target Loss**	**Target Acc.**	**Ep.**	**Avg Acc.**	**Ep.**
0.10	1E-02	5E-04	1.7364	4.0122	0.4539	9	0.1244	2.0422	0.4693	8	0.4616	9
0.10	1E-02	5E-05	0.3620	4.6599	0.5392	27	0.0413	2.5762	0.3258	7	0.4325	17
0.50	1E-03	5E-05	0.2006	4.7065	0.4731	30	0.0708	2.7137	0.3405	20	0.4068	25
0.50	1E-03	5E-04	0.4624	5.5909	0.4753	27	0.0214	1.9969	0.3349	23	0.4051	25
0.20	5E-03	5E-05	1.8225	10.4933	0.4501	11	0.0495	3.0678	0.3263	6	0.3882	9

Table 7: DANN adaptation on $P \to C$ and $P \to S$, with dynamic α

			$P \to S$				$P \to C$					
α_*	$\mathbf{lr}_0$	**wd**	**Source Loss**	**Target Loss**	**Target Acc.**	**Ep.**	**Source Loss**	**Target Loss**	**Target Acc.**	**Ep.**	**Avg Acc.**	**Ep.**
0.10	1E-02	5E-04	3.3946	6.6853	0.4172	29	0.0715	5.3923	0.4480	30	0.4326	30
0.20	5E-03	5E-05	0.0659	2.7175	0.3986	11	0.0876	5.4858	0.4539	23	0.4262	17
0.20	5E-03	5E-04	0.0730	3.7737	0.2268	6	0.6583	6.6863	0.5320	28	0.3794	17
1.00	1E-03	5E-04	0.0652	1.9615	0.2950	25	0.5018	2.5440	0.4339	11	0.3644	18
0.70	1E-03	5E-05	0.0642	2.8887	0.1861	29	0.3120	2.9795	0.4821	25	0.3341	27

Table 8: DANN adaptation on $P \to C$ and $P \to S$, with dynamic α and learning rate schedule

Tables 7 and 8 contain the best five results obtained grid search the respectively by using and not using the learning rate schedule, but making use in both cases of a dynamic value of α. The complete set of results are in Tables 12 and 13.
With the fixed learning rate the model achieves an average accuracy of 46.16% in a low number of epochs, with an initial learning rate equal to $1 \cdot 10^{-2}$ and $\alpha_* = 0.10$.
Adding the use of the learning rate scheduler, the model achieves an average accuracy of 43.26% with $lr_0 = 1 \cdot 10^{-2}$ and $\alpha_* = 0.10$.
Using these hyperparameters two models are trained on $P \to A$ task. Table 9 contains the results obtained with them. The model that does not use the learning rate scheduler achieves 53.52% of accuracy on Art Painting dataset. This result is comparable with the best model using a fixed α from Section 3.2.2.
The model which uses both the dynamic α and the LR scheduler achieves 54.83% of accuracy on the test set.

α_*	$\mathbf{lr}_0$	**wd**	**Epochs**	**LR Schedule.**	**Source Loss**	**Target Loss**	$P \to A$ **Acc.**
0.10	$1 \cdot 10^{-2}$	$5 \cdot 10^{-4}$	9	No	0.3282	3.5523	0.5352
0.10	$1 \cdot 10^{-2}$	$5 \cdot 10^{-4}$	30	Yes	0.0953	3.7655	0.5483

Table 9: DANN for domain adaptation on $P \to A$

References

[1] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-Adversarial Training of Neural Networks, 2016. http://arxiv.org/abs/1505.07818 arXiv:1505.07818.

[2] Li, Da and Yang, Yongxin and Song, Yi-Zhe and Hospedales, Timothy. Deeper, Broader and Artier Domain Generalization. In *International Conference on Computer Vision*, 2017.

[3] Sinno Jialin Pan, Qiang Yang. A Survey on Transfer Learning. 2009. URL: http://www.cse.ust.hk/~qyang/Docs/2009/tkde_transfer_learning.pdf.

4 Appendix

α	**Learning rate**	**Weight Decay**	**DANN Loss**	**Target Loss**	$P \rightarrow A$ **Accuracy**
0.2	$1 \cdot 10^{-3}$	$5 \cdot 10^{-4}$	0.1704	2.9606	0.5215
		$5 \cdot 10^{-5}$	0.1695	2.9744	0.5293
	$5 \cdot 10^{-3}$	$5 \cdot 10^{-4}$	0.6496	2.6185	0.5605
		$5 \cdot 10^{-5}$	0.5428	2.4632	0.5347
	$1 \cdot 10^{-2}$	$5 \cdot 10^{-4}$	1.4014	2.6212	0.4478
		$5 \cdot 10^{-5}$	1.7936	3.2170	0.4170
0.15	$1 \cdot 10^{-3}$	$5 \cdot 10^{-4}$	0.1567	2.5345	0.5190
		$5 \cdot 10^{-5}$	0.1187	2.8011	0.5156
	$5 \cdot 10^{-3}$	$5 \cdot 10^{-4}$	0.3557	3.3548	0.5396
		$5 \cdot 10^{-5}$	0.3323	4.1666	0.5376
	$1 \cdot 10^{-2}$	$5 \cdot 10^{-4}$	0.6113	3.5508	0.5327
		$5 \cdot 10^{-5}$	0.5990	2.9334	0.5352
0.1	$1 \cdot 10^{-3}$	$5 \cdot 10^{-4}$	0.2171	2.2272	0.5034
		$5 \cdot 10^{-5}$	0.1115	2.6726	0.5127
	$5 \cdot 10^{-3}$	$5 \cdot 10^{-4}$	0.0827	3.5133	0.5469
		$5 \cdot 10^{-5}$	0.1094	3.8737	0.5601
	$1 \cdot 10^{-2}$	$5 \cdot 10^{-4}$	0.3568	3.6623	0.5303
		$5 \cdot 10^{-5}$	0.5717	2.2223	0.5376
0.05	$1 \cdot 10^{-3}$	$5 \cdot 10^{-4}$	0.1212	2.5001	0.4976
		$5 \cdot 10^{-5}$	0.1026	2.8573	0.5093
	$5 \cdot 10^{-3}$	$5 \cdot 10^{-4}$	0.0322	3.4781	0.5566
		$5 \cdot 10^{-5}$	0.0464	3.8111	0.5425
	$1 \cdot 10^{-2}$	$5 \cdot 10^{-4}$	0.0550	4.4589	0.5469
		$5 \cdot 10^{-5}$	0.0929	3.0446	0.5508
0.01	$1 \cdot 10^{-3}$	$5 \cdot 10^{-4}$	0.1251	2.6373	0.4946
		$5 \cdot 10^{-5}$	0.1074	2.7941	0.5063
	$5 \cdot 10^{-3}$	$5 \cdot 10^{-4}$	0.0404	3.7653	0.5562
		$5 \cdot 10^{-5}$	0.0433	3.5332	0.5439
	$1 \cdot 10^{-2}$	$5 \cdot 10^{-4}$	0.0475	2.9170	0.5532
		$5 \cdot 10^{-5}$	0.1135	3.4157	0.5205

Table 10: AlexNet trained with DANN for $P \rightarrow A$ DA

.

			$P \to S$				$P \to C$					
α	**LR**	**WD**	**Source Loss**	**Target Loss**	**Target Acc.**	**Ep.**	**Source Loss**	**Target Loss**	**Target Acc.**	**Ep.**	**Avg Acc.**	**Ep.**
0.20	$1 \cdot 10^{-3}$	$5E-04$	0.0158	3.1123	0.2741	1	0.0545	4.0545	0.3840	26	0.3290	14
0.20	$1 \cdot 10^{-3}$	$5E-05$	0.0501	3.1786	0.2171	12	0.0623	4.6560	0.3319	30	0.2745	21
0.20	$5 \cdot 10^{-3}$	$5E-04$	0.4302	3.0279	0.2494	3	1.0263	6.7622	0.2837	2	0.2666	3
0.20	$5 \cdot 10^{-3}$	$5E-05$	2.1160	3.1878	0.2334	1	1.4913	3.1782	0.1822	1	0.2078	1
0.20	$1 \cdot 10^{-2}$	$5E-04$	9.4193	7.7689	0.0904	1	2.2627	5.3288	0.2338	1	0.1621	1
0.20	$1 \cdot 10^{-2}$	$5E-05$	1.5153	3.1809	0.1550	1	1.2052	6.0784	0.1638	1	0.1594	1
0.15	$1 \cdot 10^{-3}$	$5E-04$	0.0178	3.1444	0.2784	29	3.0329	12.1923	0.2014	30	0.2399	30
0.15	$1 \cdot 10^{-3}$	$5E-05$	0.0191	2.5390	0.2675	28	0.0426	4.9219	0.2803	28	0.2739	28
0.15	$5 \cdot 10^{-3}$	$5E-04$	0.4456	3.1672	0.3439	2	0.4932	4.7738	0.3584	2	0.3511	2
0.15	$5 \cdot 10^{-3}$	$5E-05$	0.0499	2.2534	0.3467	9	1.5051	3.1519	0.1706	1	0.2587	5
0.15	$1 \cdot 10^{-2}$	$5E-04$	6.1913	8.8046	0.0407	1	3.0329	12.1923	0.2014	1	0.1210	1
0.15	$1 \cdot 10^{-2}$	$5E-05$	5.1898	8.6141	0.0407	1	0.9009	7.7829	0.3703	3	0.2055	2
0.10	$1 \cdot 10^{-3}$	$5E-04$	0.0158	2.4595	0.2380	29	0.0489	4.7470	0.2534	24	0.2457	27
0.10	$1 \cdot 10^{-3}$	$5E-05$	0.0187	2.8958	0.2863	28	0.0455	3.5246	0.3285	20	0.3074	24
0.10	$5 \cdot 10^{-3}$	$5E-04$	0.0604	2.4181	0.3403	8	0.0998	6.5344	0.5158	30	0.4280	19
0.10	$5 \cdot 10^{-3}$	$5E-05$	0.1012	4.6303	0.1993	7	0.0757	5.7767	0.4817	26	0.3405	17
0.10	$1 \cdot 10^{-2}$	$5E-04$	0.2807	3.7789	0.2647	2	3.1621	14.8958	0.2671	2	0.2659	2
0.10	$1 \cdot 10^{-2}$	$5E-05$	3.4475	8.0025	0.0407	1	1.6603	14.9958	0.2526	2	0.1466	2
0.05	$1 \cdot 10^{-3}$	$5E-04$	0.0173	3.4255	0.2110	30	2.0203	1.9075	0.2688	30	0.2399	30
0.05	$1 \cdot 10^{-3}$	$5E-05$	0.0167	3.0002	0.2443	30	0.0410	5.2206	0.2346	28	0.2395	29
0.05	$5 \cdot 10^{-3}$	$5E-04$	0.0412	2.5300	0.3398	7	0.0125	6.1219	0.3537	30	0.3467	19
0.05	$5 \cdot 10^{-3}$	$5E-05$	0.0125	2.8249	0.3148	12	0.0135	5.9904	0.3938	30	0.3543	21
0.05	$1 \cdot 10^{-2}$	$5E-04$	2.7891	5.3977	0.4133	22	0.0859	5.3033	0.4735	22	0.4434	22
0.05	$1 \cdot 10^{-2}$	$5E-05$	0.1401	2.0893	0.3851	5	0.0761	5.1821	0.4275	14	0.4063	10
0.01	$1 \cdot 10^{-3}$	$5E-04$	0.0171	2.3544	0.2924	30	0.0391	4.4554	0.2538	27	0.2731	29
0.01	$1 \cdot 10^{-3}$	$5E-05$	0.0345	3.2802	0.2049	18	0.0319	5.0074	0.2410	30	0.2230	24
0.01	$5 \cdot 10^{-3}$	$5E-04$	0.0043	2.9243	0.3123	20	0.0306	5.9293	0.3038	22	0.3080	21
0.01	$5 \cdot 10^{-3}$	$5E-05$	0.0090	3.3676	0.3016	16	0.0218	6.1313	0.3050	19	0.3033	18
0.01	$1 \cdot 10^{-2}$	$5E-04$	0.0268	3.1546	0.3329	13	0.2426	3.6174	0.3370	5	0.3350	9
0.01	$1 \cdot 10^{-2}$	$5E-05$	0.0149	2.9617	0.3500	18	0.0657	3.6403	0.3810	10	0.3655	14

Table 11: DANN on $P \to C$ and $P \to S$

			$P \rightarrow S$				$P \rightarrow C$					
α	**LR**	**WD**	**Source Loss**	**Target Loss**	**Target Acc.**	**Ep.**	**Source Loss**	**Target Loss**	**Target Acc.**	**Ep.**	**Avg Acc.**	**Ep.**
0.01	1E-03	5E-04	0.0180	2.6947	0.2339	25	0.0354	4.5412	0.2543	28	0.2441	27
0.01	1E-03	5E-05	0.0148	2.4149	0.2810	27	0.0569	4.2745	0.2619	21	0.2715	24
0.01	5E-03	5E-04	0.0038	4.0531	0.3271	25	0.0432	5.4621	0.3345	14	0.3308	20
0.01	5E-03	5E-05	0.0086	3.0729	0.3052	12	0.0724	4.3229	0.3238	7	0.3145	10
0.01	1E-02	5E-04	0.0050	4.9370	0.3179	14	0.0584	5.2585	0.3895	10	0.3537	12
0.01	1E-02	5E-05	0.0016	3.8801	0.3342	28	0.0397	5.5262	0.3319	12	0.3330	20
0.10	1E-03	5E-04	0.0270	3.0931	0.1911	18	0.0431	4.6301	0.2794	30	0.2353	24
0.10	1E-03	5E-05	0.0249	2.4994	0.2825	24	0.0442	5.1566	0.2517	29	0.2671	27
0.10	5E-03	5E-04	0.0508	3.9745	0.3355	21	0.0474	4.2819	0.4168	23	0.3761	22
0.10	5E-03	5E-05	0.0190	3.2028	0.3169	12	0.0348	5.3094	0.4296	19	0.3732	16
0.10	1E-02	5E-04	0.1244	2.0422	0.4693	4	1.7364	4.0122	0.4539	9	0.4616	7
0.10	1E-02	5E-05	0.0413	2.5762	0.3258	7	0.3620	4.6599	0.5392	27	0.4325	17
0.20	1E-03	5E-04	0.1535	2.4477	0.2751	4	0.0517	4.8471	0.2786	28	0.2769	16
0.20	1E-03	5E-05	0.0300	2.9851	0.2372	19	0.0445	4.6479	0.3067	30	0.2720	25
0.20	5E-03	5E-04	0.1655	4.9245	0.2479	4	0.4909	4.3384	0.4710	9	0.3594	7
0.20	5E-03	5E-05	0.0495	3.0678	0.3263	6	1.8225	10.4933	0.4501	11	0.3882	9
0.50	1E-03	5E-04	0.0214	1.9969	0.3349	23	0.4624	5.5909	0.4753	27	0.4051	25
0.50	1E-03	5E-05	0.0708	2.7137	0.3405	20	0.2006	4.7065	0.4731	30	0.4068	25
0.70	1E-03	5E-04	0.0715	2.9378	0.2446	23	0.2438	3.0791	0.4548	12	0.3497	18

Table 12: DANN with dynamic value of α and no LR scheduler for domain adaptation on $P \rightarrow C$ and $P \rightarrow S$

			$P \to S$				$P \to C$					
α	**LR**	**WD**	**Source Loss**	**Target Loss**	**Target Acc.**	**Ep.**	**Source Loss**	**Target Loss**	**Target Acc.**	**Ep.**	**Avg Acc.**	**Ep.**
0.01	1E-03	5E-05	0.0632	2.6628	0.1911	30	0.1063	4.0417	0.2176	29	0.2044	30
0.01	5E-03	5E-05	0.0131	2.8170	0.2820	14	0.0628	5.6344	0.2432	14	0.2626	14
0.01	1E-02	5E-04	0.0649	2.5928	0.3502	5	0.0285	5.5590	0.3174	20	0.3338	13
0.10	1E-03	5E-04	0.0588	2.8531	0.2077	28	0.1193	3.7859	0.2312	22	0.2195	25
0.10	1E-03	5E-05	0.0614	1.8912	0.3665	26	0.1065	3.5044	0.2351	29	0.3008	28
0.10	5E-03	5E-04	0.0292	3.9710	0.2405	11	0.0342	6.0259	0.2756	29	0.2581	20
0.10	1E-02	5E-04	3.3946	6.6853	0.4172	29	0.0715	5.3923	0.4480	30	0.4326	30
0.10	1E-02	5E-05	0.0868	2.3540	0.3383	4	0.0232	6.1748	0.3234	26	0.3308	15
0.20	1E-03	5E-04	0.0569	2.5535	0.2247	29	0.1073	3.5741	0.2555	26	0.2401	28
0.20	1E-03	5E-05	0.0522	2.3987	0.2334	30	0.1303	3.1382	0.2628	17	0.2481	24
0.20	5E-03	5E-04	0.0730	3.7737	0.2268	6	0.6583	6.6863	0.5320	28	0.3794	17
0.20	5E-03	5E-05	0.0659	2.7175	0.3986	11	0.0876	5.4858	0.4539	23	0.4262	17
0.20	1E-02	5E-04	0.1489	5.3397	0.2507	4	1.2652	8.9047	0.3191	8	0.2849	6
0.50	1E-03	5E-04	0.0578	2.7418	0.2153	29	0.1645	3.0106	0.3439	30	0.2796	30
0.50	1E-03	5E-05	0.0604	2.8108	0.2293	29	0.1117	3.7066	0.2564	25	0.2429	27
0.70	1E-03	5E-04	0.0630	2.5938	0.2016	28	0.1399	3.8176	0.2713	30	0.2365	29
0.70	1E-03	5E-05	0.0642	2.8887	0.1861	29	0.3120	2.9795	0.4821	25	0.3341	27
1.00	1E-03	5E-04	0.0652	1.9615	0.2950	25	0.5018	2.5440	0.4339	11	0.3644	18
1.00	1E-03	5E-05	0.0806	2.4820	0.2054	26	0.5137	2.6601	0.4019	10	0.3036	18

Table 13: DANN with dynamic value of α and LR scheduler for domain adaptation on $P \to C$ and $P \to S$