Evaluation of Deep Neural Network Domain Adaptation Techniques for Image Recognition

Venkata Santosh Sai Ramireddy Muthireddy
MSc Autonomous Systems
Hochschule Bonn-Rhein-Sieg
Bonn, Germany
santosh.muthireddy@smail.inf.h-brs.de

Alan Preciado Grijalva
MSc Autonomous Systems
Hochschule Bonn-Rhein-Sieg
Bonn, Germany
alan.preciado@smail.inf.h-brs.de

***Abstract*—It has been well proved that deep networks are efficient at extracting features from a given (source) labeled dataset. However, it is not always the case that they can generalize well to other (target) datasets which very often have a different underlying distribution. In this report, we evaluate four different domain adaptation techniques for image classification tasks. The selected domain adaptation techniques are unsupervised techniques where the target dataset will not carry any labels during training phase. The experiments are conducted on the office-31 dataset. Link to github repository: https://github.com/agrija9/Deep-Unsupervised-Domain-Adaptation.**

***Index Terms*—domain adaptation, unsupervised learning, convolutional neural networks, transfer learning**

Introduction

Deep neural networks based methods have been producing state-of-the-art (SOTA) results for many problems in machine learning and computer vision. These methods require large amount of training and testing data to achieve the expected result. Although the model is trained with large datasets sometimes it will not generalize learned knowledge to new environment and datasets. This is because deep learning algorithms assume that training and testing data is drawn from independent and identical distributions (i.i.d.). However this assumption rarely holds true, as there will be a shift in data distributions across different domains this is explained in Fig. 1. This domain shift between source and target datasets will make deep neural networks produce wrong predictions on the target dataset. So training of a deep neural network with source and target datasets which reduces the domain shift between distribution of datasets is called as **domain adaptation** [1].

There are different types of domain adaptation (DA) techniques, a few of them are *Unsupervised DA* [2] [3], *Semi Supervised DA* [4], *Weakly Supervised DA* [5], *One Shot DA* [6], *Few Shot DA* [7], *Zero Shot DA* [8]. In this report, we discussed on Unsupervised DA techniques. Unsupervised DA is called so because unlabeled target dataset is used during training to reduce the domain shift.

Further Unsupervised DA have different types based on the technique used to reduce the domain shift between the source and target datasets. They are:

- Adversarial Methods [2]
- Distance-based Methods [3] [9]
- Incremental Methods [10]
- Optimal Transport [11]
- Other Methods [12]

Identify applicable funding agency here. If none, delete this.

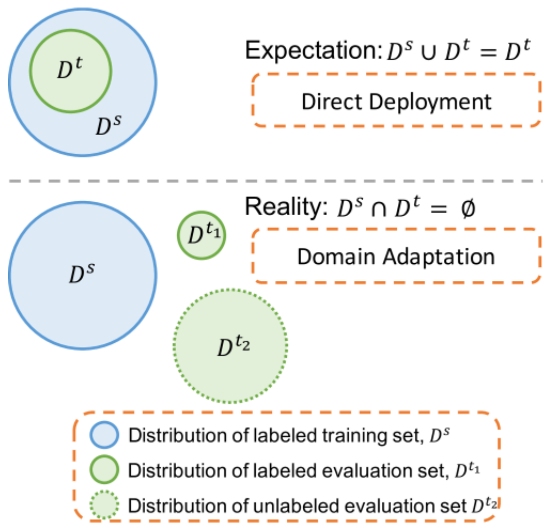

Expectation: $D^s \cup D^t = D^t$
Direct Deployment
D^t
D^s
Reality: $D^s \cap D^t = \emptyset$
Domain Adaptation
D^s
D^{t_1}
D^{t_2}
Distribution of labeled training set, D^s
Distribution of labeled evaluation set, D^{t_1}
Distribution of unlabeled evaluation set D^{t_2}

Fig. 1: Domain adaptation in the true data space: Expectation vs. Reality. [13]

In this report, techniques evaluated are selected from Adversarial and Distance-based methods. In Adversarial methods, Conditional adversarial domain adaptation (CDAN) and CDAN with Entropy conditioning (CDAN+E) [2] are selected. In Distance-based methods, Deep domain confusion: Maximizing for domain invariance (DDC) [9] and Deep coral: Correlation alignment for deep domain adaptation [3] are selected. Application of domain adaptation techniques can be used in different types of domain shift explained in Fig. 5 and application in dataset to dataset domain shift is selected for evaluation. In this report evaluation of different domain adaptation techniques are benchmarked on office-31 dataset [14].

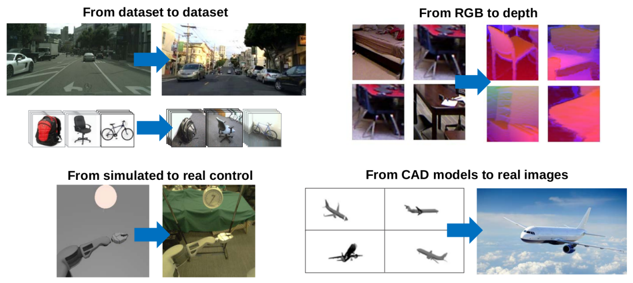

From dataset to dataset
From RGB to depth
From simulated to real control
From CAD models to real images

Fig. 2: Applications to different types of domain shift [15]

Domain Adaptation Methods

DeepCORAL

DeepCORAL [3] is a domain adaptation method based on a previously proposed method called Correlation Alignment (CORAL) [16].

CORAL attempts to increase performance in domain shift by aligning the second order statistics between both data distributions. This method consists of 1) feature extraction, 2) apply a linear transformation and 3) implement an SVM classifier.

DeepCORAL extends the capacities of CORAL by learning a non-linear mapping using convolutional neural networks and a differentiable loss function that attempts to minimize the Frobenius norm between the feature covariance matrices of source and target distributions. This method works in an unsupervised way where the target dataset has no labels.

Recall that the main goal is to learn representations that allow to achieve good performance in both data distributions (image recognition, for example). To achieve this, DeepCORAL intializes the network parameters from pre-trained AlexNet model while at the same time minimizing the loss mentioned above (*a.k.a* CORAL loss). The architecture of DeepCORAL is shown in the figure below.

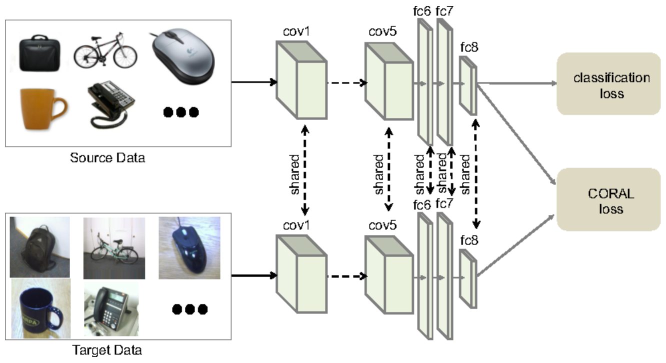


Fig. 3: DeepCORAL neural network architecture.

Note that DeepCORAL takes the last fully connected layer of AlexNet (*fc8*) and computes a classification and CORAL loss.

CORAL loss: This loss is defined for a single layer feature map. In this case for *fc8*, although not mentioned, we believe the authors run a grid search to identify the optimal layer within the network to compute the loss. This can have to do with the task at hand.

To formalize the CORAL loss, suppose that we have source-domain training examples $D_s = \{x_i\}$, where $x \in \mathbb{R}^d$ with labels $L_s = \{y_i\}$, i $\in \{1, ...L\}$, and unlabeled data $D_T = \{u_i\}$, $u \in \mathbb{R}^d$. Let n_s and n_t be the number of source and target data available respectively. Recall that **x** and **u** are the d-dimensional deep layer activations (*fc8*) $\phi(I)$ of an input image I. Let C_S and C_T be the feature covariance matrices of these activations for source and target domains respectively. The CORAL loss is defined as the difference between these covariances (second order statistics):

$$l_{CORAL} = \frac{1}{4d^2}\|C_S - C_T\|_F^2 \tag{1}$$

where $\|\cdot\|_F^2$ refers to the squared matrix Frobenius norm. The covariance matrices C_S, C_T are defined as

$$C_S = \frac{1}{n_S - 1}\big(D_S^T D_S - \frac{1}{n_S}(1^T D_S)^T(1^T D_S)\big) \tag{2}$$

$$C_T = \frac{1}{n_T - 1}\big(D_T^T D_T - \frac{1}{n_T}(1^T D_T)^T(1^T D_T)\big) \tag{3}$$

where **1** is a column vector with all elements equal to 1.

The gradient of the CORAL loss can be calculated using the chain rule. Refer to the paper for the analytical expression.

In practice, we do batch learning and thus the covariances are computed over batches. Moreover, the network parameters are shared between the two networks.

In the context of multi-class classification, the total loss that is used to train is a joint term between with classification loss (cross-entropy) and CORAL loss. The first one trying to optimize classification accuracy and the second trying to learn features that can work well in both data domains. The total loss is

$$l = l_{CLASS} + \sum_{i=1}^{t} \lambda_i l_{CORAL} \tag{4}$$

here t is the number of CORAL loss computed per layers. In this case it is a single layer. The parameter λ is introduced as a regularizer to help reach an equilibrium between these two losses at the end of training.

Deep Domain Confusion (DDC)

Deep domain confusion is another domain adaptation method that uses a deep neural network to learn a non-linear transformation. It is similar to DeepCORAL in the sense that it introduces a transfer loss to learn features for both data domains. [9]

In their paper, Tzeng et al. show the concept of biased datasets and how trained classifiers don't always transfer well to target domains. Their approach is to maximize a loss function that minimizes classification error and maximizes domain confusion.

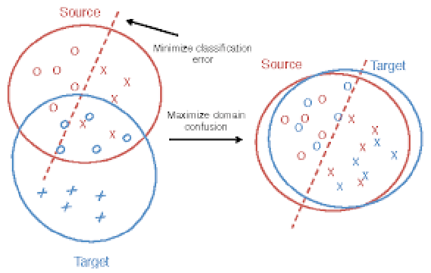


Fig. 4: Deep Domain Confusion approach: Maximizing domain confusion can lead to good performance in target domains.

To minimize the distance between data distributions, DDC uses a standard distribution distance metric called Maximum Mean Discrepancy (MMD). In this case, this distance is computed with respect to an activation layer $\phi(\cdot)$ acting on source data points $x_S \in X_S$ and target data points $x_T \in X_T$. The approximation to this distance is as follows

$$MMD(X_S, X_T) = \left| \frac{1}{\|X_S\|} \sum \phi(x_s) - \frac{1}{\|X_T\|} \sum \phi(x_t) \right| \quad (5)$$

Training a model with MMD and a classification loss will result in a strong classifier ready to transfer accross domains. Thus, the total loss is

$$l = l_C(X_L, y) + \lambda MMD^2(X_S, X_T) \quad (6)$$

Here l_C corresponds to a classification loss on the available source labelled data X_L and the ground truths y. The hyperparameter λ acts in a similar way a regularizer as in DeepCORAL.

The architecture proposed in DDC is similar to DeepCORAL; it intializes its weights with a pre-trained AlexNet and weights are shared between both data domains. The difference is that it has an extra *bottleneck* layer (

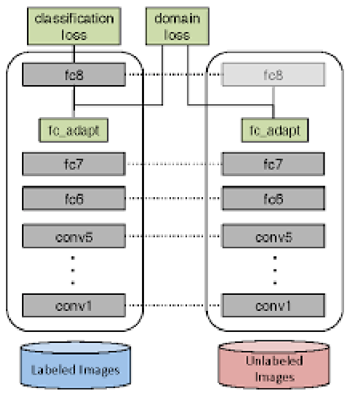


Fig. 5: Deep Domain Confusion approach: Maximizing domain confusion can lead to good performance in target domains.

CDAN

Adversarial domain adaptation [17] [18] [19] consists of domain adaptation and adversarial learning which is very similar to Generative Adversarial Networks (GANs) [20]. In adversarial domain adaptation, a deep neural network model learns image representations by minimizing a classification loss and simultaneously a domain discriminator learns how to distinguish between source and target domain. There are however two pitfalls for adversarial domain adaptation methods. In the first place they cannot handle datasets which have complex multi modal distributions. And secondly we cannot rely on the uncertain information difference between source and target data to model the domain discriminator.

These two pitfalls can be solved by conditional domain adaptation techniques. Conditional domain adaptation is made possible due to the research carried out in Conditional Generative Adversarial Networks (CGANs) [21]. CGANs proved a point that domain shift between distributions of real and synthetic images can be reduced by conditioning network using discriminative information.

CDAN [2] is an adversarial domain adaptation technique motivated by CGANs and it makes use of the discriminative information obtained from a deep classifier network. The key to CDAN models is a new conditional domain discriminator conditioned to the covariance of representations of domain-specific resources and classifier predictions. Let us consider $D_s = \{(x_i^s, y_i^s)\}_{i=1^s}^n$ is a source domain data with n_s labeled examples and $D_t = \{(x_j^t)\}_{j=1^t}^n$ is a target domain dataset with n_t unlabeled examples. Source and target datasets are drawn from $P(x^s, y^s)$ and $Q(x^t, y^t)$ which are not i.i.d. i.e., $P \neq Q$. The aim of CDAN is to design a network $G : x \to y$ that can reduce the domain shift between source and target datasets. $f = F(x)$ and $g = G(x)$ are feature representation and classifier predictions made from the network G respectively.

Considering all the things mentioned above, CDAN is formalized as a minmax optimization problem. In minmax optimization, two terms are considered as players, namely $E(G)$ corresponding to source classifier G and minimized to guarantee lower source risk and $E(D, G)$ on G and the domain discriminator D on cross domains. $E(D, G)$ is minimized over D and maximized over f and g

$$E(G) = \mathbb{E}_{(x_i^s, y_i^s) \sim D_s} L(G(x_i^s), y_i^s) \quad (7)$$

$$E(D, G) = -\mathbb{E}_{x_i^s \sim D_s} log[D(f_i^s, g_i^s)] - \mathbb{E}_{x_i^t \sim D_t} log[D(f_j^t, g_j^t)] \quad (8)$$

where L(.) is the cross entropy loss. Now CDAN is described in minmax optimization problem as

$$\min_{\text{G}} E(G) - \lambda E(D, G) \; and \min_{\text{G}} E(D, G) \quad (9)$$

where λ is a hyper-parameter that balances the domain adversary and source risk.

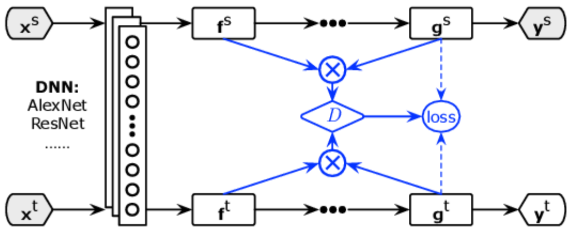


Fig. 6: Architecture of CDAN for domain adaptation [2]

In Fig. 6 D is the domain discriminator conditioned using a multilinear map $f \otimes g$, multilinear mapping is explained in Section 3.2 of [2]. Thus CDAN enables domain adaptation over cross domains using minmax optimization.

CDAN+E

CDAN+E is an variant of CDAN. In CDAN+E we will have additional involvement of **Entropy Conditioning** which improves the difference between source and target domains. In CDAN, the domain discriminator allots equal weight to each and every example irrespective of examples with uncertain predictions from G. So uncertainity corresponding to classifier predictions are quantified using entropy criteria explained in Eq. 10.

$$H(g) = -\sum_{c=1}^{C} g_c log\ g_c \tag{10}$$

where C is number of classes and g_c is probability of predictions.

Now training examples are prioritized by an entropy-aware weight given by

$$w(H(g)) = 1 + e^{-H(g)} \tag{11}$$

So CDAN+E minmax optimization is formulated as

$$\begin{aligned}
&\underset{G}{\min} \mathbb{E}_{(x_i^s, y_i^s) \sim D_s} L(G(x_i^s), y_i^s) \\
&\quad + \lambda(\mathbb{E}_{x_i \sim D_s} w(H(g_i^s)) log[D(T(h_i^s))] \\
&\quad + \mathbb{E}_{x_j^t \sim D_t} w(H(g_j^t)) log[1 - D(T(h_j^t))]) \\
&\quad \underset{D}{\max} \mathbb{E}_{x_i \sim D_s} w(H(g_i^s)) log[D(T(h_i^s))] \\
&\quad + \mathbb{E}_{x_j^t \sim D_t} w(H(g_j^t)) log[1 - D(T(h_j^t))]
\end{aligned} \tag{12}$$

where h =(f,g) is a joint variable.

Thus CDAN+E encourages the certain predictions which are achieved through entropy minimization principle [22].

Experiments

In this section we describe the experiments performed with the different metods on different data domains. Our implementations are written in Pytorch and are located in the link to the repository given above. The experiments were carried out using Google Colab GPU support (Testla P4).

Office-31 dataset

Office-31 dataset was first introduced in [14]. It contains a total of 4652 images collected from three different sources. The three different sources from which images are collected are online web (amazon), digital SLR (dslr) camera and webcam. Office-31 dataset have 31 classes [1] in total.

- **Images from the web:** These are collected from amazon website. We call this dataset as amazon dataset (A). It has around 2800 images and with an average of 90 images per each class. The resolution of the images are each 300x300 pixels.
- **Images from a dslr camera:** These are captured from dslr camera in real environments. We call this dataset as dslr dataset (D). It has total of 498 images and with an average of 16 images per each class. The resolution of the images are 1000x1000 pixels.
- **Images from a webcam:** These are collected from a webcam with lower resolution and will have more noise. From now this is called as webcam dataset (W). It has around 795 images and with an average of 25 images per each class. The resolution of the images is HxW where H and W are varying between 400 and 600.

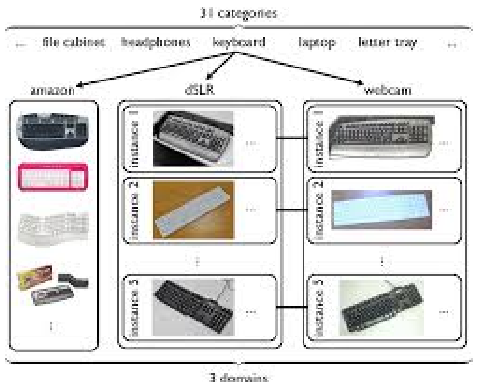


Fig. 7: Sample of keyboard category images for all the three domains [14]

DeepCORAL

The method is evaluated with the benchmark Office dataset described above. We have carried out 6 different experiments for all the domain shifts available (given the tree datastes cotained in Office).

CORAL loss is applied to the last fully connected layer (fc8). The weights of this layer are initialized following a normal distribution $\mathcal{N}(0, 0.005)$ and the weights from the other layers are initialized with the pre-trained AlexNet. We used Stochastic gradient descent (SGD).

DeepCORAL was trained using batches of 128 images (both domains), a fixed learning rate of 10^{-3}, weight decay set to

[1] The 31 classes are: backpack, bike, bike helmet, bookcase, bottle, calculator, desk chair, desk lamp, computer, file cabinet, headphones, keyboard, laptop, letter tray, mobile phone, monitor, mouse, mug, notebook, pen, phone, printer, projector, puncher, ring binder, ruler, scissors, speaker, stapler, tape, and trash can.

$5x10^{-4}$ and momentum to 0.9. The λ factor is set in such a way that both losses are approximately the same. In practice, given t epoch iterations, λ is $\frac{1}{t}$. Furthermore, during training only the source dataset is labelled, this is how we compute the classification loss.

In Table 1 we show the obtained accuracies for each method for all 6 experiments. These values correspond to the image classification accuracy on the target dataset. From here, we see that DeepCORAL obtained the best result in 4 out of 6 experiments. This method outputs accuracy values in the range between 38% to 89%. The fluctuation of accuracies between domains has to do directly with the chosen source and target distributions. It is important to note that the method does not output accuracies greater than 50% in most of the cases. We've compared our values with the original paper and although lower by a small percentage, these follow the same trend as the original results.

For a better visualization of the method's performance we have added to visualize accuracy and loss as a function of epochs for the case Amazon $\rightarrow$Webcam. Figure 8a) shows an increase of almost 10% in recognition accuracy when including CORAL loss. Figure 9a) shows the behavior of the loss functions, note that they tend to become equal due to the lambda factor.

Deep Domain Confusion

Training for this method is very similar to DeepCORAL. The main differences are 1) it trains with MMD loss, 2) adds the bottleneck layer and 3) it uses a schedule learning rate which decreases as a function of epochs. Fig. 8b) shows that accuracy with MMD (or DDC) doesn't have a very noticeable increase compared to training without MMD, however, it seems to be more stable and shows an steady increase. Due to time constrains, we were only able to run one experiment for this method. However, it is consistent with the accuracy reported on the original paper.

CDAN and CDAN+E

Experiments for CDAN and CDAN+E are conducted with same set of hyperparameters and the only difference between both is the transfer loss function that is involved. Hyperparameters used in the experiments for CDAN and CDAN+E are:

- Epochs 100
- Source batch size 10
- Target batch size 10
- Learning rate 1e-3
- Momentum 0.9
- Weight decay 5e-4

Dataset used for all the experiments are office-31 dataset [14] Backbone used in this experiment is pretrained AlexNet with a small modification. There is a bottleneck layer introduced before last fully connected layer. Bottleneck layer has the dimension 256 (used in the original paper [2]) which is also a user defined shape. Optimizer used is Stochastic Gradient Descent (SGD) in which parameters of deep neural network G and domain discriminator D are added to optimize. During the training phase each batch from source and target data loader are passed through deep neural network G and features f from bottleneck layer and fully connected layer output are obtained from the deep neural network G. Thus obtained features f and final layer output of source and target datasets are joined together. Concatenated final layer outputs are passed through softmax layer in order to get network predictions g.

As we obtained features f and predictions g, classification loss for source data can be calculated using g_s and ground truth labels of source data. Transfer loss is calculated based on domain adaptation method selected. If CDAN is selected, joint variable h=(f,g) which have both source and target data features and predictions are used. Thus transfer loss is calculated using domain discriminator network whose input is h which is explained in above Sections on CDAN. If CDAN+E is selected, along with the features and predictions we calculated entropy from class predictions which is explained in above Section. CDAN+E . This calculated entropy is used to perform weighted operations based on uncertainty of the predictions. Total loss to do back propagation is calculated using below equation.

$$Total\ loss = classification\ loss + \lambda\ transfer\ loss \quad (13)$$

where λ is a hyperparameter which is described in the above Sections. During the experiment it is observed that transfer loss obtained is very small which resulted in very very small gradients of order 10^{-7}. Which in turn did not update the weights of domain discriminator and thus the transfer loss is fluctuated with very small variance. In Table I we can observe that CDAN+E only performed better in 2 cases out of 6. It is also observed that transfer loss is reducing till 20^{th}epoch in most of the cases. In Fig. 8c and 8d, training and testing accuracy as well as in Fig. 9c and 9d, transfer loss and classification loss of A$\rightarrow$W domain adaptation is shown respectively. In our opinion smaller batch size also played a key role in results of CDAN and CDAN+E methods. Smaller batch size is chosen considering the limitations of local machine. Although the results are not matched with original paper we observe that there is variation in accuracy and loss when transfer loss is included.

Problems/Challenges encountered

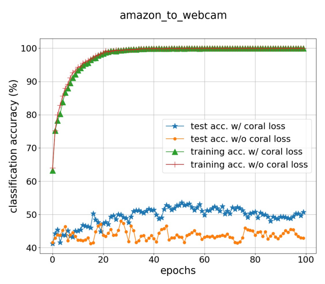
- Understanding and implementing the end-to-end pipeline in Pytorch.

- Data bottlenecks during training led us to use Google Colab or addressing a computer with higher memory capacity.

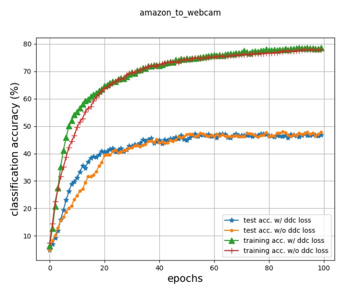
- In Adversarial domain adaptation local machine is used and batch size is limited to 10.

- In CDAN and CDAN+E, very small transfer loss is observed which resulted in very very small gradients of order 10^{-7}. Which we believe resulted in vanishing gradient problem.

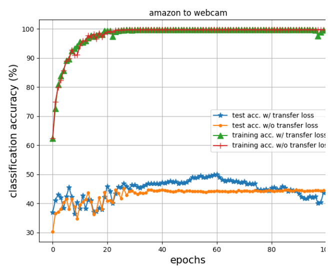
- This resulted in poor results in both the methods and original implementation of authors of [2] is also behaving the same way with same backbone and hyperparameters.



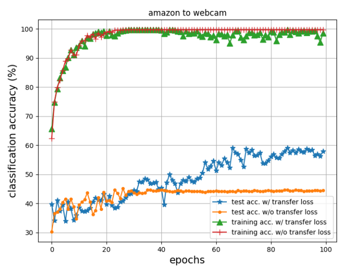
(a) DeepCORAL



(b) DDC

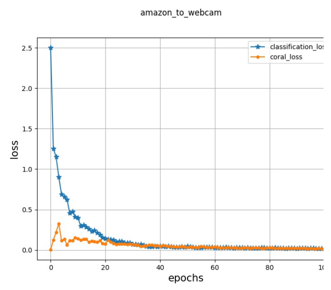


(c) CDAN

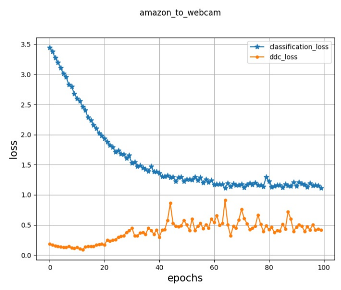


(d) CDAN+E

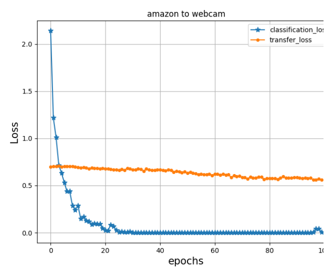
Fig. 8: Classification and transfer loss for each of the evaluated methods as a function of epochs.



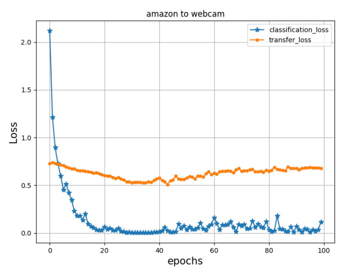
(a) DeepCORAL



(b) DDC



(c) CDAN



(d) CDAN+E

Fig. 9: Image recognition accuracies for the four evaluated methods. The domain shift in this case corresponds to Amazon to Webcam. Red and green lines correspond to the classification accuracy on the source test data. Blue and orange lines correspond to the target test data.

Conclusions

Based on the methods we evaluated, we can say there are three possible scenarios for accuracy performance. Note that in the graphs presented here, the accuracy increases when we add a domain adaptation loss (e.g. Amazon $\rightarrow$Webcam), however, we noticed that accuracy can decrease or stay the same for other domain shifts. Refer to the repository to see the other plots. Moreover, recognition accuracies don't go higher than 70% in most experiments, this means there is a lot of room for improvement. For future work we can carry out trainings for 200 epochs (not 100) and understand better the behavior of our losses, for instance why in some cases losses oscillate too much. Finally, we can conclude that we gained hands-on experience building end-to-end pipelines using deep neural networks for domain adaptation tasks.

References

[1] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, Oct 2010.

[2] M. Long, Z. CAO, J. Wang, and M. I. Jordan, "Conditional adversarial domain adaptation," in *Advances in Neural Information Processing Systems 31*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds. Curran Associates, Inc., 2018, pp. 1640–1650. [Online]. Available: http://papers.nips.cc/paper/7436-conditional-adversarial-domain-adaptation.pdf

[3] B. Sun and K. Saenko, "Deep coral: Correlation alignment for deep domain adaptation," in *Computer Vision – ECCV 2016 Workshops*, G. Hua and H. Jégou, Eds. Cham: Springer International Publishing, 2016, pp. 443–450.

[4] K. Saito, D. Kim, S. Sclaroff, T. Darrell, and K. Saenko, "Semi-supervised domain adaptation via minimax entropy," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct 2019, pp. 8049–8057.

[5] S. Tan, J. Jiao, and W. Zheng, "Weakly supervised open-set domain adaptation by dual-domain collaboration," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Los Alamitos, CA, USA: IEEE Computer Society, jun 2019, pp. 5389–5398. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/CVPR.2019.00554

[6] J. Hoffman, E. Tzeng, J. Donahue, Y. Jia, K. Saenko, and T. Darrell, "One-shot adaptation of supervised deep convolutional models," *arXiv preprint arXiv:1312.6204*, 2013.

[7] T. Wang, X. Zhang, L. Yuan, and J. Feng, "Few-shot adaptive faster r-cnn," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Los Alamitos, CA, USA: IEEE Computer Society, jun 2019, pp. 7166–7175. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/CVPR.2019.00734

[8] K.-C. Peng, Z. Wu, and J. Ernst, "Zero-shot deep domain adaptation," in *Computer Vision – ECCV 2018*, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds. Cham: Springer International Publishing, 2018, pp. 793–810.

[9] E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, and T. Darrell, "Deep domain confusion: Maximizing for domain invariance," *arXiv preprint arXiv:1412.3474*, 2014.

[10] M. Wulfmeier, A. Bewley, and I. Posner, "Incremental adversarial domain adaptation for continually changing environments," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018, pp. 4489–4495.

[11] G. O. Jimenez, M. E. Gheche, E. Simou, H. P. Maretic, and P. Frossard, "Cdot: Continuous domain adaptation using optimal transport," *arXiv preprint arXiv:1909.11448*, 2019.

[12] M. Binkowski, D. Hjelm, and A. Courville, "Batch weight for domain adaptation with mass shift," in *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[13] X. Xu, X. Zhou, R. Venkatesan, G. Swaminathan, and O. Majumder, "d-sne: Domain adaptation using stochastic neighborhood embedding," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019, pp. 2492–2501.

TABLE I: Testing accuracy for different domain adaptation techniques across different datasets.

	$A \to W$	$A \to D$	$W \to A$	$W \to D$	$D \to A$	$D \to W$
No Adaptation	43.1±2.5	49.2±3.7	35.6±0.6	94.2±3.1	35.4±0.7	90.9±2.4
DeepCORAL	49.5±2.7	40.0±3.3	38.3±0.4	74.4±4.3	38.5±1.5	89.1±4.4
DDC	41.7±9.1	—	—	—	—	—
CDAN	44.9±3.3	49.5±4.6	34.8±2.4	93.3±3.4	32.9±2.4	88.3±3.8
CDAN+E	48.7±7.5	53.7±4.7	35.3±2.7	93.6±3.4	33.9±2.2	87.7±4.0

[14] K. Saenko, B. Kulis, M. Fritz, and T. Darrell, "Adapting visual category models to new domains," in *Computer Vision – ECCV 2010*, K. Daniilidis, P. Maragos, and N. Paragios, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 213–226.

[15] B. Kulis, K. Saenko, and T. Darrell, "What you saw is not what you get: Domain adaptation using asymmetric kernel transforms," in *CVPR 2011*, June 2011, pp. 1785–1792.

[16] B. Sun, J. Feng, and K. Saenko, "Correlation alignment for unsupervised domain adaptation," *CoRR*, vol. abs/1612.01939, 2016. [Online]. Available: http://arxiv.org/abs/1612.01939

[17] Y. Ganin and V. Lempitsky, "Unsupervised domain adaptation by backpropagation," in *Proceedings of the 32nd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, F. Bach and D. Blei, Eds., vol. 37. Lille, France: PMLR, 07–09 Jul 2015, pp. 1180–1189. [Online]. Available: http://proceedings.mlr.press/v37/ganin15.html

[18] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell, "Adversarial discriminative domain adaptation," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[19] E. Tzeng, J. Hoffman, T. Darrell, and K. Saenko, "Simultaneous deep transfer across domains and tasks," in *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[20] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[21] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.

[22] Y. Grandvalet and Y. Bengio, "Semi-supervised learning by entropy minimization," in *Advances in neural information processing systems*, 2005, pp. 529–536.