6.867 MACHINE LEARNING PROJECT FINAL DOCUMENT

Maohao Shen, Charles Lyu, Zhuo Liu

1 INTRODUCTION

As mentioned in the Strategy Document, our project is based on the Co-regularized Domain Alignment (Co-DA) algorithm proposed by Kumar et al. (2018). It trains two models $\boldsymbol{\theta}_1 = \boldsymbol{h}_1 \circ \boldsymbol{f}_1$ and $\boldsymbol{\theta}_2 = \boldsymbol{h}_2 \circ \boldsymbol{f}_2$ that minimize the multiple different loss functions. Our project can be mainly summarized into two parts:

- Studying the domain adaptation problem and the original Co-DA framework, developing a deeper understanding of the algorithm and its differences from other standard approaches.
- Implementing new ideas based on the original algorithm, including incorporating pseudo labeling and using alternative domain alignment loss.

1.1 ANALYSIS OF ORIGINAL ALGORITHM

Domain adaptation problem is a well-studied but still popular research topic, so it is interesting to study one of the state-of-the-art methods. We mainly study the Co-DA algorithm from the following perspectives:

- **Visualization.** The intuition behind domain adaptation is to align the source and target distributions in the feature space during training. We visualize this using the dimensionality reduction technique taught in class. The results are shown in Section 3.1.
- **Ablation Study.** We conduct ablation study of the Co-DA algorithm to observe and explain the contribution of different loss functions and the intuition behind them. The results are shown in Section 3.2.

1.2 NEW IMPLEMENTATION

Beyond the original algorithm, we also explore and implement some new ideas:

- **Pseudo Labeling.** This is inspired from recent literature (Venkat et al., 2021) which showed self-learning methods can enforce an implicit domain alignment process, i.e. if a sufficient number of high-quality pseudo labels are provided in the target domain, simply using supervised learning methods will encourage domain alignment without auxiliary feature alignment. We generate pseudo labels on the target dataset using an ensemble of the two models trained by Co-DA, keep only the pseudo labels with high confidence, and apply supervised training by including an additional loss term on those pseudo-labeled target data in the objective. Formally:

$$P(\boldsymbol{y}|\boldsymbol{x}^t) = \frac{1}{m}\sum_{j=1}^{m} P\left(\boldsymbol{y}|\boldsymbol{x}^t;\boldsymbol{\theta}_j\right) \tag{1}$$

  Where $P(\boldsymbol{y}|\boldsymbol{x}^t)$ is the ensemble prediction of m different models; in practice, $m = 2$.

$$\tilde{\boldsymbol{y}}^t = \begin{cases} \arg\max_{\boldsymbol{y}} P(\boldsymbol{y}|\boldsymbol{x}^t) & \text{if } P(\boldsymbol{y}|\boldsymbol{x}^t) > \eta \\ \emptyset & \text{Otherwise} \end{cases}$$

  Where η is the confidence threshold. Only target data with high confidence $P(\boldsymbol{y}|\boldsymbol{x}^t)$ will be assigned pseudo labels $\tilde{\boldsymbol{y}}^t$, and for these pseudo-labeled target data, we use cross-entropy loss for supervised training. We denote the pseudo labelled target data as $\mathcal{D}_l$ and the

remaining unlabeled target data as $\mathcal{D}_u$, and $\mathcal{D}_t = \mathcal{D}_l \cup \mathcal{D}_u$. Thus, formally, we add an additional loss into the original objective:

$$\mathcal{L}_{\text{pl}} = \lambda_{\text{pl}} \cdot -\mathbb{E}_{\boldsymbol{x}^t, \tilde{\boldsymbol{y}}^t \in \mathcal{D}_l} \left[(\tilde{\boldsymbol{y}}^t)^T \log \left(\boldsymbol{\theta}_j \left(\boldsymbol{x}^t \right) \right) \right] \tag{2}$$

The pseudo labels are updated before each new epoch. The training steps are shown in Algorithm 1, and results using pseudo labeling are shown in Section 3.3.

Algorithm 1: Algorithm using pseudo labeling

Input: Multiple models $\{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, ..., \boldsymbol{\theta}_m\} = \{(\boldsymbol{h}_1 \circ \boldsymbol{f}_1), (\boldsymbol{h}_2 \circ \boldsymbol{f}_2), ..., (\boldsymbol{h}_m \circ \boldsymbol{f}_m)\}$; target domain unlabeled data $\mathcal{D}_t := \{\boldsymbol{x}_i^t\}_{i=1}^{n_t}$, maximum iterations T

1 **for** $\tau = 1, 2, \ldots, T$ **do**

2 For each target data point $\boldsymbol{x}_i^t$, generate pseudo label using equation 1.

3 Update $\mathcal{D}_l$ and $\mathcal{D}_u$: if $\tilde{\boldsymbol{y}}_i^t \neq \emptyset$, $\mathcal{D}_l = \mathcal{D}_l \cup \boldsymbol{x}_i^t$, otherwise $\mathcal{D}_u = \mathcal{D}_u \cup \boldsymbol{x}_i^t$

4 Retrain each model $\boldsymbol{\theta}_j$ by optimizing loss with additional cross entropy loss on pseudo-labeled target data shown in equation 2.

5 **end**

Return: optimal models $\{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, ..., \boldsymbol{\theta}_m\}$

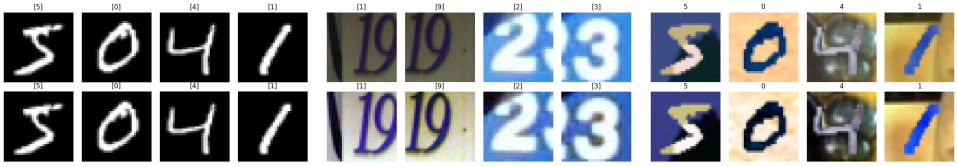
- **MMD Loss.** As shown in our strategy document, the original Co-DA algorithm trains a domain discriminator to enforce feature alignment. It's also interesting to see if other alternative domain alignment losses may also achieve the same goal. We use the MMD loss (Gretton et al., 2012) as a divergence-based loss, and compare it with the original adversarial loss in Co-DA. The empirical results using MMD loss are shown in 3.4.

2 Experiment Setup

2.1 Dataset and Models

Datasets

We use three datasets for our experiments: MNIST, SVHN and MNIST-M, which are also used in Kumar et al. (2018). MNIST is a dataset of grey-scale hand-written digits, and MNIST-M combines the digits in MNIST with random color patches from the BSDS500 dataset. SVHN is a real-world dataset of house numbers from street views. These datasets are illustrated in Figure 1.



(a) MNIST (b) SVHN (c) MNIST-M

Figure 1: Samples from MNIST, SVHN and MNIST-M datasets. Top row are the original images, and bottom row are instance normalized images. MNIST images only have one channel, so instance normalization does not change the images.

Model architecture

Our network is the same as Co-DA. We train two neural networks each consisting of three parts: a feature generator $\boldsymbol{f}$, a classifier $\boldsymbol{h}$, and a domain discriminator d. The two nets have separate parameters but the same architecture, and trained using a single objective function.

The feature generator consists of two groups of 3 convolutional layers (64 channels), each group followed by a max pooling layer (kernel size 2, stride 2), dropout ($p = 0.5$) and random noise. The classifier consists of 3 convolutional layers (64 channels), a global pool and a fully connected layer into 10 features. Each convolutional layer is followed by batch norm and leaky ReLU. The discriminator has a fully connected layer with 1000 features, a ReLU, and a final fully connected layer that outputs a single value.

2.2 Implementation Details

Instance normalization

Instance normalization applies the operation $\ell\left(x^{(i)}\right) = \frac{x^{(i)} - \mu\left(x^{(i)}\right)}{\sigma\left(x^{(i)}\right)}$,where $x^{(i)} \in \mathbb{R}^{H \times W \times C}$ denotes the i^{th} sample with height H, width W and C channels. μ and σ are functions that compute the mean and standard deviation across the spatial dimensions (leaving the channel dimension). One notable property of instance normalization is that it is invariant to channel-wide scaling and shifting: for $\gamma, \beta \in \mathbb{R}^C$, $\gamma > 0$,$\ell\left(x^{(i)}\right) = \ell\left(\gamma x^{(i)} + \beta\right)$. Applying instance normalization to the input layer makes the classifier invariant to different shifts and scaling of the pixel intensities between different channels, which may help reducing the extent to which small changes to the hypothesis in the source domain can lead to large changes in target domain. The visualization of instance normalization is shown in the second row of Figure 1.

Learning algorithms and hyperparameters

After some tuning, the hyper parameters we found to have the best performance are ($\lambda_c = 5.0, \lambda_d = 1.0, \lambda_{sv} = 1.0, \lambda_{ce} = 10^{-2}, \lambda_p = 0.1, \lambda_{div} = 10^{-2}, \nu = 10$). The hyper-parameter η in equation 1 is simply the mean of confidence across all target data. We apply Adam Optimizer Kingma & Ba (2017) (learning rate $= 0.001; \beta_1 = 0.5; \beta_2 = 0.999$) with an exponential moving average (EMA) with momentum $\beta = 0.998$ to the parameter trajectory:

$$\theta^{n+1} = \text{Adam}(lr, \beta_1, \beta_2)(\theta^n),\ \beta_1 = 0.5, \beta_2 = 0.999, \text{lr} = 0.001$$
$$\theta^{n+1}_{EMA} = (1 - \beta)\theta^{n+1} + \beta\theta^n_{EMA},\ \beta = 0.998$$

While the Adam optimizer modifies θ in every mini-batch (the batch size is 64), the validation and test process are based on θ_{EMA} instead of θ. The minimization of domain loss requires solving a mini-max optimization problem. We instead seek the minimization of two separate losses with two optimizers (one for optimizing the domain discriminator and one for minimizing the domain loss). We use the training data from both source and target domain for training, and report the results as the accuracy among all the testing data from target domain after 20 epoch (20k iterations).

3 Experiment Results and Analysis

3.1 Results and Visualization

We performed four experiments transforming MNIST and SVHN to each other, as well as MNIST and MNIST-M. For each task, we run three different training methods: Only training on the source dataset without domain adaptation; adaptation without co-regularization, using the the Virtual Adversarial Domain Adaptation (VADA) model by Shu et al. (2018) which Co-DA is based on; and Co-DA.

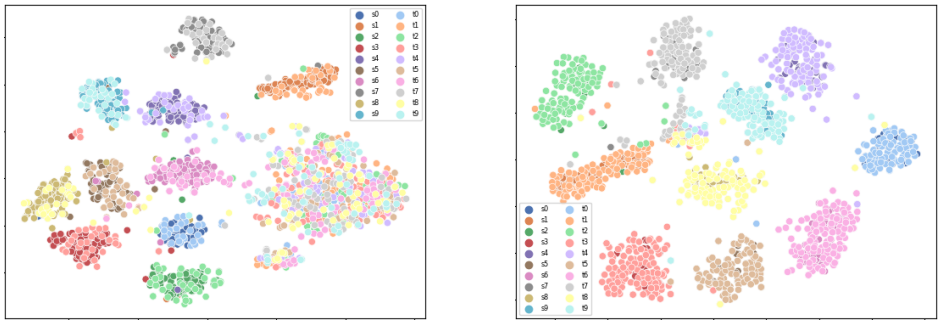
Source	**Method**	**M→ S**	**S→ M**	**M→ MM**	**MM→ M**
No Adaptation	source model	33.66	77.72	40.42	99.11
Adapted	VADA	64.21	78.52	96.44	99.50
	Co-DA	72.46	84.37	97.33	99.46

Table 1: **Target accuracies (%) of Co-DA and other methods on select experiments:** M, S, and MM are abbreviations of MNIST, SVHN, and MNIST-M. *Source model* denotes training the model only on the source dataset, and then applying it on target dataset for testing.

Table 1 shows the experimental results. While all methods achieve high accuracy on source data (not shown), domain adaptation methods (VADA and Co-DA) have higher accuracies on the target dataset, especially for the more challenging MNIST → SVHN and MNIST → MNIST-M tasks. Compared to the source model, Co-DA improves the accuracy of the former task by 38.8%, and the latter by 56.91%.

Figure 2 illustrates the improvement by comparing the alignment of source and target domains in the feature space in the MNIST → MNIST-M task. Figure 2a shows a large number of target data points

in the bottom right cluster, which have not been aligned to their respective source domain clusters. Figure 2b shows Co-DA's feature extractor aligning most of the target domain with the source domain with the correct label, even though target labels are not observed during training.

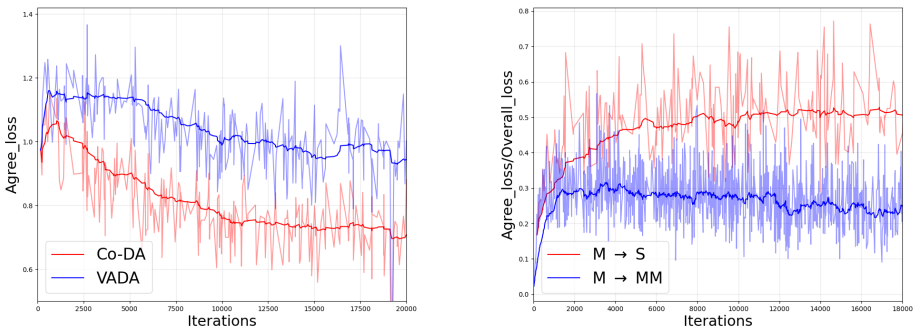


(a) Train on source only (40.42% accuracy) (b) Co-DA (97.33% accuracy)

Figure 2: t-SNE plots of feature spaces mapped from source and target domains for MNIST→MNIST-M, after training using two different methods without and with domain adaptation. Dark-colored dots (s0∼s9) are source features, light-colored dots (t0∼t9) are target features. Colors indicate true labels.

The difference is much smaller for the SVHN → MNIST and MNIST-M → MNIST tasks, where Co-DA only improves by 6.65% and 0.35% over the source model. We think this is due to the complexities of the datasets. For example, MNIST-M is adapted from MNIST with color patches, so a source model trained on MNIST-M is expressive enough to also perform well on the simpler MNIST dataset. On the other hand, a model only trained on MNIST would be less likely to distinguish colors, background noise and transformations of digits that are prevalent in SVHN and MNIST-M. In other words, domain adaptation is most useful when transforming from a simpler distribution to a more complex one.

3.2 Ablation Study



(a) (b)

Figure 3: (a) Agreement loss ($\lambda_p L_p(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2; P_t)$) comparison between Co-DA and VADA for MNIST → SVHN task. (b) Agreement loss divided by overall loss ($\lambda_p L_p(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2; P_t)/L$) comparison between MNIST → SVHN task and MNIST → MNIST-M task when using Co-DA.

We also test the model when the agreement loss $L_p(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2; P_t)$ and diverse loss $D_{\boldsymbol{f}}(\boldsymbol{f}_1, \boldsymbol{f}_2)$ are set to zero, which recovers the VADA model. From Table 1, it can be seen that Co-DA achieves overall higher accuracies than VADA, especially for the most challenging task MNIST → SVHN, where Co-DA improves the accuracy by $\sim 8.2\%$. One significant problem with VADA is that the class conditional distribution of the target domain is not guaranteed to be aligned even if the overall marginal distribution is aligned well, because we do not have access to the labels of target domain (Intuitively, it can be thought of as the situation where the distribution of class i in the target domain is aligned with the distribution of class j in the source domain, while $i \neq j$). Encouraging the two

predictors to agree with each other effectively reduces the probability of the case where overall distributions are aligned but expected error is high. Figure 3a shows Co-DA leads to significantly lower agreement loss than VADA.

However, we also observe that agreement loss is not significant on MNIST-M → MNIST and MNIST → MNIST-M tasks. We think that this is because the similarity of these two datasets lets the class conditional distribution also align well after domain alignment, which makes the agreement loss comparably low. Figure 3b compares the agreement loss component between the two tasks. Indeed the MNIST → MNIST-M has significant lower agreement loss component, which makes Co-DA outperforms VADA less significantly in this task.

3.3 Results using Pseudo labels

As shown in Table 2, it can be seen that adding pseudo label in training can further improve the performance on tasks SVHN → MNIST and MNIST → MNIST-M, achieve comparable results on task MNIST-M → MNIST, but will lead to worse performance on the challenging task MNIST → SVHN. Based on our experiment, we find that the quality of pseudo label will greatly affect the performance: intuitively, if the pseudo label itself has a lot of noise in it, it will induce bias in training and lead to worse performance. For example, for the challenging task MNIST → SVHN, our pseudo label has poor quality and thus the results are worse than Co-DA without pseudo labeling. However, we also find that good pseudo labels result in faster convergence and serve as calibration for domain alignment.

Source	**Method**	**M→ S**	**S→ M**	**M→ MM**	**MM→ M**
No Adaptation	source model	33.66	77.72	40.42	99.11
Adapted	Co-DA	72.46	84.37	97.33	99.46
	Co-DA+Pseudo Label	47.33	96.54	98.11	99.44

Table 2: **Results of using pseudo label:** M, S, and MM are abbreviations of MNIST, SVHN, and MNIST-M. *Source model* denotes training the model only on the source dataset, and then applying it on target dataset for testing.

3.4 Results using different alignment loss

As shown in Table 3, the MMD loss achieves comparable results on some tasks, but perform worse than Co-DA on others. It's hard to tell which alignment loss is better in theory, but we conjecture that the difference between the MMD loss and the original approach using domain discriminator lies in the scope of the training data used. The MMD loss is optimized over a small batch of data during training, whose distribution may be different from that of the whole dataset. However, the domain discriminator is learned and optimized over the entire dataset, so it might have more robust performance.

Source	**Method**	**M→ S**	**S→ M**	**M→ MM**	**MM→ M**
No Adaptation	source model	33.66	77.72	40.42	99.11
Adapted	Co-DA	72.46	84.37	97.33	99.46
	Co-DA+MMD Loss	38.16	91.97	86.71	99.47

Table 3: **Results of using MMD Loss:** M, S, and MM are abbreviations of MNIST, SVHN, and MNIST-M. *Source model* denotes training the model only on the source dataset, and then applying it on target dataset for testing.

4 Discussion

We studied one of the state-of-the-art unsupervised domain adaptation algorithms not covered in class. Through this project, we gained deeper understanding of domain adaptation, and more importantly, we also had a hands-on opportunity to apply the techniques taught in class to this practical research problem, such as dimensionality reduction, adversarial training, and anything relevant to training a deep neural network. Domain adaptation is still a intriguing and popular research topic, so we believe this project may also inspire our own research in the future.

5 TEAM CONTRIBUTION

In general, all three members had roughly equal efforts in different parts in this project, including brainstorming, experiments, and report write-up.

Maohao Shen picked the topic for this project, mainly proposed the new ideas and implemented the corresponding experiments, including the pseudo label and alternative domain alignment loss.

Charles Lyu worked on adapting the data loader from the original code to import all datasets, generating the t-SNE plots that visualize domain alignment, and producing experimental results involving the MNIST-M dataset.

Zhuo Liu mainly worked on debugging the original code and producing the experimental results on MNIST and SVHN datasets, including parameter tuning and ablation study.

REFERENCES

Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *The Journal of Machine Learning Research*, 13(1):723–773, 2012.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

Abhishek Kumar, Prasanna Sattigeri, Kahini Wadhawan, Leonid Karlinsky, Rogerio Feris, William T Freeman, and Gregory Wornell. Co-regularized alignment for unsupervised domain adaptation. *arXiv preprint arXiv:1811.05443*, 2018.

Rui Shu, Hung H. Bui, Hirokazu Narui, and Stefano Ermon. A dirt-t approach to unsupervised domain adaptation, 2018.

Naveen Venkat, Jogendra Nath Kundu, Durgesh Kumar Singh, Ambareesh Revanur, and R Venkatesh Babu. Your classifier can secretly suffice multi-source domain adaptation. *arXiv preprint arXiv:2103.11169*, 2021.