Semi-Supervised Domain Adaptation with Auto-Encoder via Simultaneous Learning (Supplementary Material)

Md Mahmudur Rahman
University of Massachusetts Lowell
mdmahmudur_rahman@student.uml.edu

Rameswar Panda
MIT-IBM Watson AI Lab
rpanda@ibm.com

Mohammad Arif Ul Alam
University of Massachusetts Lowell
mohammadariful_alam@uml.edu

This supplementary material contains the following sections:

- Section 1: How does our method work?
- Section 2: Implementation Details.
- Section 3: Training Algorithm.
- Section 4: Visual Examples

1. How does our method work?

In this section, we will discuss the theoretical basics why our auto-encoder based domain adaptation framework works and explain its main advantage over discrepancy based methods on classifier stream network. The secret recipe lies in the bottleneck feature representation of the auto-encoders which is stated using three theorems as follows: First, we will provide an theoretical explanation and then we will explain intuitively.

The following theorem characterizes the theoretical base of our proposed framework. Here we will provide a theoretical background of how our system work with motivation from the work of Qian et al. [4]. Then we will provide an intuitive explanation.

Considering our auto-encoder based learning framework depicted in equation 1, 2 and figure 2 in the paper, we can consider the following assumption:

Assumption 1 : The bottleneck layer feature embedding distribution of the same classes in the source and the target domains are same. Formally, if $C_s = C_t$ then, $E_s(X^s_{C_s}) = E_t(X^t_{C_t})$ where C_s and C_t are the corresponding class labels of source and target domain data.

Assumption 2 : The bottleneck feature distribution of different classes are different. If $C_s \neq C_t$ then, $E_s(X^s_{C_s}) \neq E_t(X^t_{C_t})$. It can be combined with the first part.

Theorem 1 : Let, X_C is the set of common feature samples of both source and target domains of class C. If the above assumptions hold, For each class C, there exist a class conditioned domain encoder $E^*(X_C)$ and a decoder $D^*(E^*(X_C))$, and

$$\lim_{X_C \to \infty} \frac{1}{X_C} \mathcal{L}_{cw\_MMD}(P_{\hat{X}_s \to_t}(\cdot|C,V), P_{X_C}(\cdot|C,V)) = 0 \quad (1)$$

Here $P_{\hat{X}_s \to_t}(\cdot|C,V)$ is the conditional probability of feature embedding adaptation from the source to target with respect to the class C and the domain invariant features V. The star (*) notation on the encoder and the decoder indicates to include both domains. What about proof of this theorem. If this motivated from some paper then we should mention the paper here.

The Theorem 1 can be interpreted as follows. If the number of samples in X_C is high enough to hold the distribution matching property and if the bottleneck dimension is properly set, the optimization loss functions presented in equation 3 and 4 will satisfy the ideal domain adaptation property presented in equation 10. This ideal domain adaptation will match the class-wise feature space distribution between the source and the target domains. However, if the auto-encoder bottleneck dimension of the source and the target domain is optimal, it will only hold the domain invariant features V_C for class C and discard the class-wise domain specific features U^s_C and U^t_C. This selection procedure of domain invariant features V_C in the bottleneck layers of auto-encoders enables effective feature distribution matching between corresponding classes of source and target domains.

As stated earlier, almost all of the discrepancy based domain adaptation frameworks align the feature space at some layers in the middle of a classification network.

In figure 1 we show intuitively how the feature representation space of our auto-encoder framework differs from the feature representation space of the classification stream network. In classification stream network, the samples of same class cluster together because it is an intermediate stage of an classification network. Because of clustering most of the samples in the feature representation space $\hat{X}$ of the same class lose one to one relationship with the input features X. As a result, the domain adaptation with the discrepancy based methods in this feature space only works with the centroids of the clusters of different classes rather than with the individual samples. This can result to negative transfer when there are heterogeneity present with the class labels.

On the other hand, the representation space of the bottleneck layer of deep auto-encoder is not clustered. The feature representation space is scattered because of the nature of the of the optimization function. One to one relationship also hold between the input features X and the feature representation space $\hat{X}$. As a result, during the domain adaptation optimization, transfer of features from the source to the target domain occurs all over the regions of feature representation space rather than just in the clusters of classes. This way domain adaptation with auto-encoder can achieve state-of-the-art performance using a simple training scheme.

2. Implementation Details

We keep the symmetry of the network architecture of the auto-encoders along the bottleneck layer as shown in figure 3 in the paper. As a result, we follow the topology of the encoder layers similarly to the decoder layers just in reverse direction. We replace the maxpool layers in the corresponding decoder modules with upsampling layers to keep the symmetric structure. We use a dense layer of 100 neurons with relu activation as the bottleneck layer. In order to match the output shape and size with the shape and size of the input sample, we added an output convolution layer as the final layer of the decoder segment. The final output layer has the number of filters same as the number of channels in the images with a filter size of 3×3. To implement the simultaneous learning with both of the auto-encoder, we also use a dummy layer between the bottleneck layers of two auto-encoders which maintain the source and the target auto-encoder in a single graph. The dummy layer does not affect on weight updates as it consists of a unity weight with zero activation function.

3. Training Algorithm

Algorithm 1: Training Procedures of Auto-encoder based Semi-Supervised Domain Adaptation (AutoDA)

Input : Source Domain Labeled data, $\mathcal{D}^s = \{X^s, Y^s\}$, Target Domain Labeled data, $\mathcal{D}_l^t = \{X_l^t, Y_l^t\}$, Unlabeled Target Domain, $\mathcal{D}_u^t = \{X_u^t\}$, model parameter β, classifier layers size c_l, and bottleneck feature space size b

Output: End to end trained classifier prediction network for target domain unlabeled data

Match the number of samples class by class between $\mathcal{D}^s$ and $\mathcal{D}_l^t$ by randomly resampling the smaller domain;

Sort $\mathcal{D}^s$ and $\mathcal{D}_l^t$ by class. Initialize the layer weights of the source and the target auto-encoder randomly;

Set the corresponding loss functions $\mathcal{L}_s$ and $\mathcal{L}_t$ to the source and the target auto-encoder respectively;

repeat

  Update weights of the source and target auto-encoders with batches of data from $\{X^s, X^s\}$ and $\{X_l^t, X_l^t\}$ respectively.

until *$\mathcal{L}_s$ and $\mathcal{L}_t$ converges*;

Take only Encoder module of source Auto-encoder network and cascade the Classifier network at the end of it;

Freeze the Encoder weights and randomly initialize the weights of the Classifier network;

repeat

  Optimize the "Source Encoder + Classifier" network with X^s, Y^s;

until *Validation Loss converge*;

Take The Classifier module and cascade with the target encoder;

Freeze the target encoder weights;

repeat

  Optimize the "Target Encoder + Classifier" network with labelled target data, $\{X_l^t, Y_l^t\}$;

until *Validation loss converge*;

Estimate the label of the unlabeled samples from target domain, $\mathcal{D}_u^t = \{X_u^t\}$ with "Target Encoder + Classifier" network;

4. Visual Examples

References

[1] Peng Gao, Jingmei Li, Guodong Zhao, and Changhong Ding. Multisource deep transfer learning based on bal-

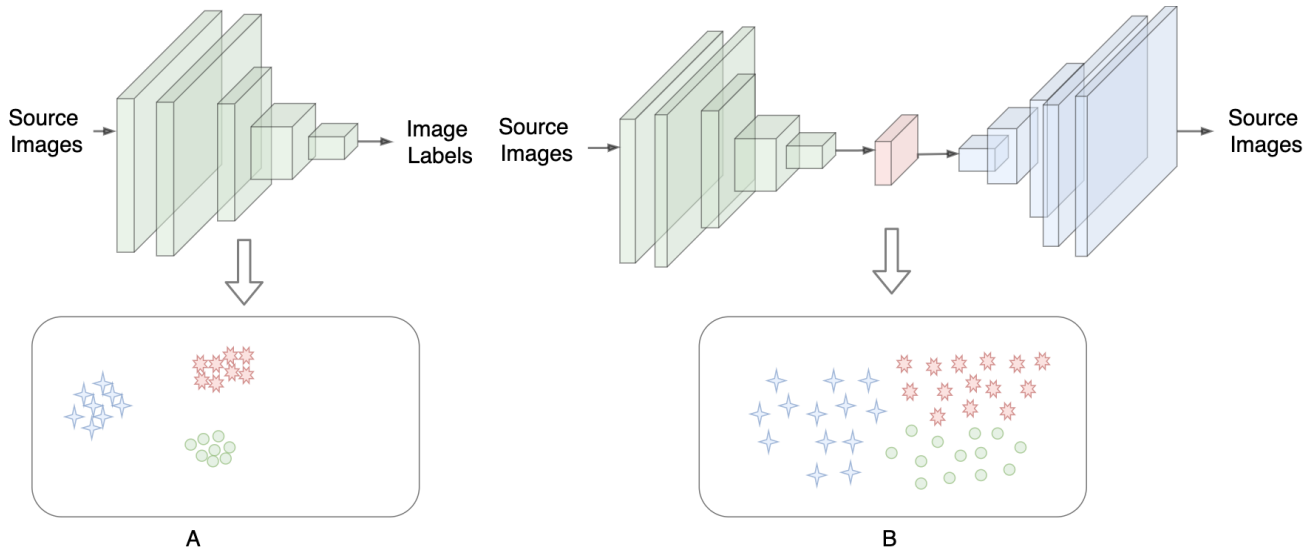

Figure 1: Feature representations space of a. Classification stream network, b. Auto-encoder network.

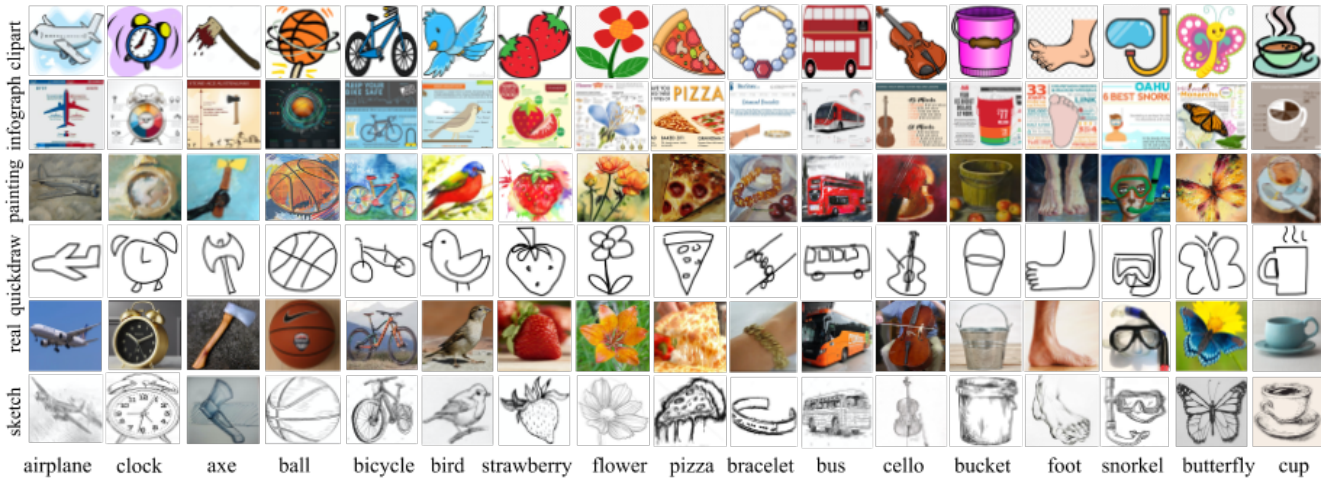

Figure 2: Visual examples from the DomainNet dataset [3]

anced distribution adaptation. *Computational Intelligence and Neuroscience*, 2022, 2022.

[2] Zhihai He, Bo Yang, Chaoxian Chen, Qilin Mu, and Zesong Li. Clda: an adversarial unsupervised domain adaptation method with classifier-level adaptation. *Multimedia Tools and Applications*, 79(45):33973–33991, 2020.

[3] Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1406–1415, 2019.

[4] Kaizhi Qian, Yang Zhang, Shiyu Chang, Xuesong Yang, and Mark Hasegawa-Johnson. Autovc: Zero-shot voice style transfer with only autoencoder loss. In *International Conference on Machine Learning*, pages 5210–5219. PMLR, 2019.

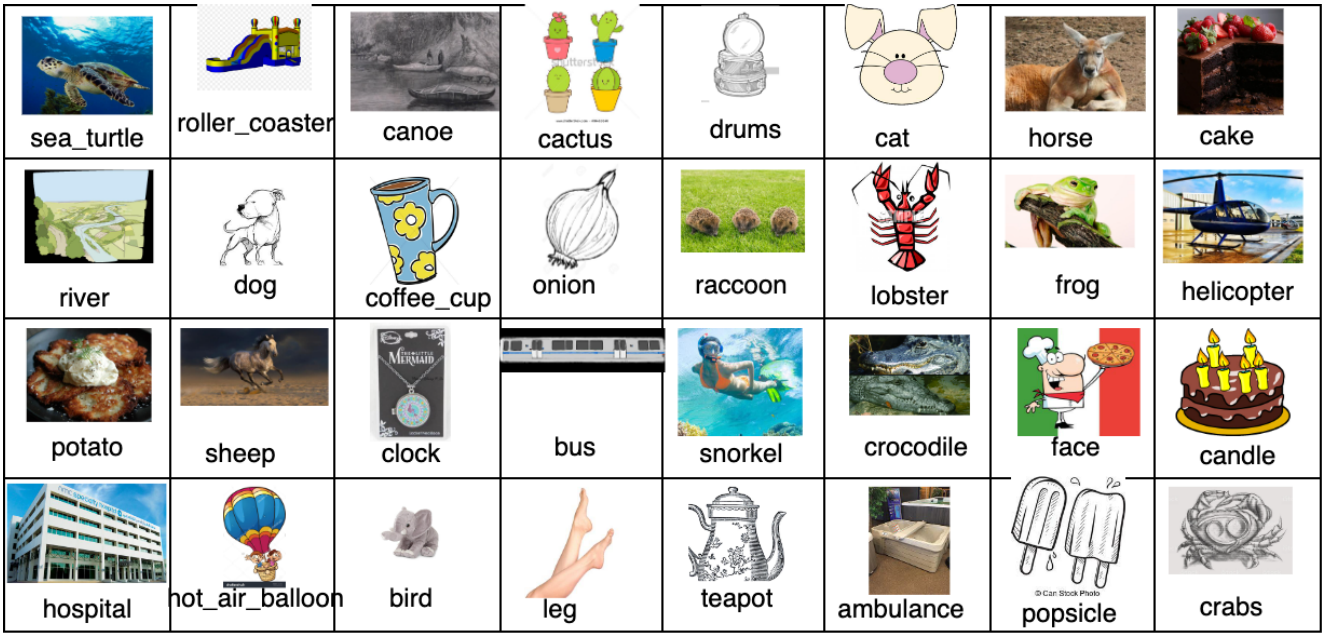

Figure 3: Visual examples of predictions of our model from the DomainNet dataset

Figure 4: Some visual examples from the Office31 dataset [2]

Figure 5: Some visual examples of predictions of our model from the Office31 dataset

Figure 6: Visual examples of different digit recognition datasets [1]