Drift-Aware Predictive Coding for Adaptation in Changing Environments

Haozhu Wang, Zhangxing Bian, Rui Guo, Anthony Liang, Mingyu Yang
University of Michigan, Ann Arbor, MI, USA
{hzwang,zxbian,guorui,aliangdw,mingyuy}@umich.edu

Abstract

In many scenarios, dataset shifts continuously over time and that leads to decayed performance of machine learning models. Existing domain adaptation methods either rely on learning representations that are invariant to the dataset shift, or gradually adapt to the shifting domains step by step. However, these methods neglect the temporal ordering of the collected data, and it could lead to suboptimal performance when the underlying domain shift has a learnable dynamics. In this work, we propose a drift-aware predicting coding scheme that is able to represent temporally ordered domains as a sequence of embedding vectors. With the awareness of embedding vectors from past domains, we could have a better adaption to the future domains where the labels are missing. Experiments on two toy datasets and two benchmarks (Rotating MNIST and a Portaits dataset) demonstrate that the proposed method outperforms the baselines without learning the dynamics of domain shifting.

1. Introduction

Machine learning models are often deployed for an extended period of time, e.g., several weeks or a season. However, dataset shift is common for real-life applications, e.g, self-driving, chemical sensors, etc [2, 12]. Frequently labeling new data and updating the model are expensive and sensitive to noisy data. *Domain adaptation* tackles the dataset shift problem by learning representations that are invariant to domains shift [9, 15, 13, 19, 11]. However, most domain adaptation methods neglect the temporal ordering of collected data. Thus, they cannot leverage the underlying domain shift dynamics to better adapt to the shifting environment. Recently, researchers proposed to use self-training to gradually adapt a model trained on a labeled source domain to a target domain, with the aid of multiple temporally ordered intermediate domains [12]. However, the pseudo-labels generated for self-training may be inaccurate when dataset shift is large, leading to poor domain adaptation performance. In addition, the self-training approach does not directly learn the domain shift dynamics, which may lead to inferior performance than a method that directly models the shift dynamics.

We argue that domain shifting dynamics is informative for making predictions in a gradually shifting environment. In this project, we propose to directly learn the environment shift dynamics through representation learning. For each domain, our proposed approach learns a **D**rift-**A**ware **P**redictive(DAP) coding, which embeds information helpful for predicting the next domain,and modulates the base learner(for example, a classifier in a classification problem) to adapt to the shifted new domain. Our method is end-to-end trainable and during training, we sample each domains to build a chain connecting all domains. During testing, in each step, we collect the predictive embedding from last domain and modulate the base learner with that embedding. We show that this approach can capture the shifting dynamics between domains and adjust decision boundaries in each step. Meanwhile, we also show that utilizing unlabelled target domain data can promote the learning process.

To summarize, the main work of this project are: **(1)** We propose to directly learn the shifting dynamics in a gradually shifting environment and show that these dynamics are informative for making predictions. **(2)** We improve the learned shifting dynamics by exposing unlabelled target domain data through adversarial learning. **(3)** We show the changing of decision boundaries on synthesized data and show quantitative results on Rotating-MNIST and Yearbook Portrait[7] data.

The remainder of this paper is organized as follows. A brief summary of related work is provided in Sec.2 . Details of training and the pose prediction process are explained in Sec.3. Experimental results are reported in Sec.4 and Sec. 5. The paper concludes in Sec.6.

2. Related Work

Unsupervised Domain adaptation In unsupervised domain adaptation, the goal is to directly adapt from a labeled source domain to an unlabeled target domain. Various prior literature have been proposed including discrepancy-based methods [4], adversary based methods

[16], and reconstruction-based methods [3]. Discrepancy-based methods aim to induce alignment between the source and target domains in some feature space. One popular discrepancy measure is the maximum mean discrepancy (MMD) [4] or the distance between the mean of two domains in some kernel space. Another way to define discrepancy is by training an adversarial discriminator [16] that distinguishes between two domains. However, both these approaches, MMD and adversarial training, are very difficult to optimize in training and often fail to converge and get stuck on a local minimum. Another class of methods directly transform the source images to resemble the target images with generative models [3]. These methods operate directly on image pixels rather than the latent representation space unlike discrepancy based approaches. In this project, we use adversarial methods to promote the learning of domain shift dynamics by aligning the feature distribution after feature fusion of DAP embedding and adapt the discriminator to a continuously indexed domain setting.

Incremental Domain Adaptation Closely related to our work are the incremental domain adaptation problems, where the domain is assumed to shift smoothly over time and the goal is to adapt the source domain to multiple target domains incrementally. Different methods for traditional static domain adaptations are applied to perform pair-wise domain adaptation, such as optimal transport [10], adversarial loss[1], generative adversarial networks[18] and linear transform[8]. Recently, self-training is shown to work well for domain shifts with small Wasserstein-infinity distance, both theoretically and empirically [12]. Different from other domain adaptation methods where the goal is to match the distribution of feature vectors, self-training directly makes use of the classifier from the previous domain to provide pseudo labels for current domain, with which the classifier is fine-tuned afterwards. Here, the main difference between our approach and the methods above is that our method explicitly represents the dynamics of the domain shift and directly uses them in both training and testing.

3. Method

3.1. Overview

In this section, we formalize the problem of gradual domain shifting and describe our method to solve this problem. We first introduce the Drift-Aware-Predictive(DAP) coding and then we introduce an adversarial learning method to improve the efficacy of this method.

Problem Setting. In this paper, we are solving an unsupervised domain adaptation classification problem where we have a sequence of N source domains $\{\mathcal{D}_0, \mathcal{D}_1, \ldots, \mathcal{D}_{N-1}\}$, and a sequence of M target domains $\{\mathcal{D}_N, \mathcal{D}_{N+1}, \ldots, \mathcal{D}_{N+M-1}\}$. The source domains are all labeled while the testing domains are not. Starting from the first domain $\mathcal{D}_0$, the dataset is gradually shifting. We assume that the underlying domain shift is not completely random and has some learnable drift dynamics. With the data and labels from the source domains, our goal is to predict the labels for the target domains.

3.2. Drift-Aware Predictive Coding

The basic DAP consists of three modules: Feature extraction module, feature fusion module and task specific module. In the feature extraction module, similar with *Domain2Vec* [14], we train a feature extraction network E that extract the domain-specific vector $e_s^{i,j}$ and domain-predictive vector $e_p^{i,j}$ for the ith data from jth domain, denoted as $x^{i,j}$. That is,

$$e_s^{i,j}, e_p^{i,j} = F(x^{i,j}) \tag{1}$$

The domain-specific vector $e_s^{i,j}$ contains information that is useful for making accurate predictions on the current domain $\mathcal{D}_j$. In the mean time, the domain-predictive vector $e_p^{i,j}$ captures the necessary information from the current domain that is most helpful for the classification of the next domain $\mathcal{D}_{j+1}$. Next, in the feature fusion module, we train a feature fusion network F that fuses both the domain-specific vectors from current domain $\mathcal{D}_j$ and the domain-predictive vectors from the previous domain $\mathcal{D}_{j-1}$. To represent the predictive information from the previous domain, we take the average of all domain-predictive vectors from $\mathcal{D}_{j-1}$ as the predictive embedding vector e_p^{j-1} for $\mathcal{D}_{j-1}$. That is,

$$e_p^{j-1} = \frac{1}{|\mathcal{D}_j|} \sum_{i=0}^{|\mathcal{D}_{j-1}|-1} e_p^{i,j-1} \tag{2}$$

After that, we concatenate e_p^{j-1} with every domain-specific vector $e_s^{i,j}$ and feed them to the feature fusion network to get the fused feature vectors $z^{i,j}$, which is,

$$z^{i,j} = F(e_p^{j-1}, e_s^{i,j}) \tag{3}$$

For domain $\mathcal{D}_0$ that doesn't have any domain-predictive vectors from the previous domain, we replace it with some special hand-craft vectors. So far, we use zero vectors as e_p^{-1}. At the end, we define the problem in the task-specific module. In our case, we are solving a classification problem, and we simply pass the fused features $z^{i,j}$ to a classification network C for predictions.

Drift-Aware Predictive Loss For domain j, we define the Drift-Aware Predictive Loss as:

$$\mathcal{L}_{DAP}^{j} = \sum_{i=0}^{|\mathcal{D}_j|-1} \mathcal{L}_{CE}\left(C(e_c^{i,j}), y^{i,j}\right) \tag{4}$$

where $y^{i,j}$ denotes the ground truth class label for ith data from domain $\mathcal{D}_j$. Note that we need the ground

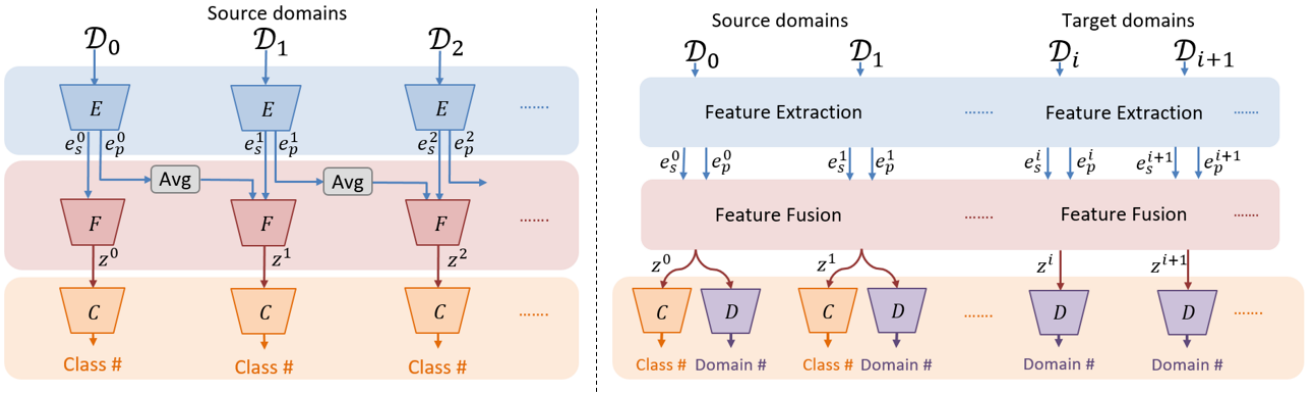

Figure 1. General structure of the proposed method. Left: The Vanilla DAP which contains three modules: Feature Extraction Module (blue), Feature Fusion module (red) and task specific module (orange). The feature extraction module extracts the domain-specific vectors e_s and drift-predictive vectors e_p; the feature fusion module fuses the domain-specific vectors and the drift-predictive vectors from the previous domain; the task specific module defines the task we would like to solve. In this paper, we are solving classification problems. Right: DAP with distribution alignment. Another discriminator (green) is added to match the fused feature vectors z for all domains, including the test domains, while the feature extraction module and feature fusion module stay the same.

truth labels to calculate the Drift-Aware Predictive Loss. Thus, we can only calculate the loss for all source domains $\{\mathcal{D}_0, \mathcal{D}_1, \ldots, \mathcal{D}_{N-1}\}$. And the total Drift-Aware Predictive Loss can be represented as their sum, i.e.,

$$\mathcal{L}_{DAP} = \sum_{j=0}^{N-1} \mathcal{L}_{DAP}^{j} \tag{5}$$

If the above loss can be successfully minimized, the domain-predictive embedding will be able to improve the next base learner before any data is collected from the shifted domain, i.e., proactively adapting to the changing environment. Since we only receive the domain-predictive vectors from the previous domain, our adaptation pipeline forms a first-order Markov chain. However, for more complicated domain shift dynamics, a higher-order might be necessary.

3.3. Aligning Feature Distribution with Adversarial Training

The basic DAP is supervised to capture the gradual domain shifting within source domains. However, without any exposure to target domains, the model may not be able to adapt well to future target domains especially when the data dimension is high. Then, the distribution of fused vectors z in the future domains may not align with the one from source domains, causing the failure of the final classification network. As a result, motivated by [17], we adopt an adversarial training mechanism to align the distribution of fused vectors from both source domains and target domains such that the final classification network could deal with the future target domains as well.

To align the fused vectors z after the feature extraction module and feature fusion module from all $N + M$ source and target domains, we require that $p(z|u_i) = p(z|u_j), \forall u_i, u_j \in \mathcal{U}$. $\mathcal{U}$ is the set of all domain indices. In our case, $\mathcal{U} = \{0, 1, 2, ..., N + M - 1\}$. This requirement implies that $p(z|u) = p(z)$ and thus z and u are independent. Similar with many adversarial approaches [5][13], we train a discriminator D to predict each domain index while our feature extraction and fusion network aim to fool the discriminator, as shown in Figure 1(right). The intuition is that, the fused vectors from all domains should be aligned if no one could successfully predict their domain indexes.

In our problem setting, the data is gradually shifting and our domain shift is continuous and ordinal. Therefore, instead of doing classification, similar with [17], our discriminator regresses the domain index. For simplicity, we use $\mathcal{L}_2$ loss and the discriminative loss for domain $\mathcal{D}_j$ can be expressed as ,

$$\mathcal{L}_d^j = \sum_{i=0}^{|\mathcal{D}_j|-1} (D(z^{i,j}) - u^j)^2 \tag{6}$$

where u^j denotes the domain index for $\mathcal{D}^j$. Then, the new optimization objective can be written as:

$$\min_{E,F,C} \max_{D} \mathcal{L}_{DAP} - \lambda \mathcal{L}_d, \tag{7}$$

where $\mathcal{L}_{DAP}$ is the loss for drift-aware predictive coding and $\mathcal{L}_d = \sum_{j=0}^{M+N-1} \mathcal{L}_d^j$. λ is a weighting term. Note that $\mathcal{L}_{DAP}$ is only calculated on the labelled source domains and $\mathcal{L}_d$ is calculated on both source and target domains.

3.4. Training Procedure

During training, in each step, we sample equal number of samples from each domain and build a chain as in Figure

1. The base learner share its weight across all domains and we modulate it with different DAP coding e_p^j to adapt it to different domains. We either use single $\mathcal{L}_{DAP}$ or the adversarial objective in 7 to train the network, depending on whether data in target domains is available.

3.5. Adapting to Test domains

After the predictive domain embedding model is trained on the sampled sequence, we can apply it for model adaptation in the testing phase. Specifically, for each new test domain $\mathcal{D}_i, \forall i \in \{N, N+1, \ldots, N+M-1\}$, we will obtain a new base learner model $f_{\theta_i|e_p^i}$ by conditioning the initial model parameters θ_0 on the drift-predictive embedding vector of the previous domain e_p^{i-1}.

4. Experiment

In this section, we experiment our Drift-Aware Predictive Domain Adaptation on three datasets, (1) Toy ellipse data and sine data (2) Rotated MNIST data and (3) American High School Portraits data(Yearbook)[7]. We show that our model with drift-aware predictive coding can still perform well during gradual domain shifting and the decision boundaries can gradually adapting according to the change of data distribution.

4.1. Synthesized Toy Data

Toy ellipse data The toy ellipse data consists of 40 domains indexed from 1 to 40, shifting from left to right as in Figure 3 first column. In all three groups of data, the distribution of left half(1st-20th domains) of the data are kept the same. Data are normally distributed normally along the boundary of a $\frac{1}{4}$ circle with radius 5. While on the right half(21st-40th domain), the data are distributed on a $\frac{1}{4}$ ellipse with fixed short axis of 5 and varying long axis $a \in \{5, 10, 15\}$. We treat this task as a binary classification problem and the decision boundary is continuously evolving during the gradual shifting.

We train on 10 domains and test on the remaining 30 domains. Note that these three experiments have training data with the same distribution in source domains while they have different data in the target domains. We use only $\mathcal{L}_{DAP}$ for training and data in target domains is not visible during training. We compare our method with a simple non-adaptation method.

Figure 3 shows the predictions of our methods and predictions of a simple baseline without any adaptation. Figure 2 shows the decision boundary of these three models under 1st,15th and 30th target domains. From these results, we show that DAP coding can capture and forecast the future trend of data even though data distributions in source domains are kept the same. DAP coding can adjust the decision boundaries during gradual domain shifting in target domains and can have dynamic decision boundaries subject to the actual domain shift.

Toy sine data The toy sine data also contains 40 domains and data points are normally distributed along a sine curve with three cycles. All the settings are the same with toy ellipse data except 15 domains are used for training. Last two rows in Figure 4 shows the predictions of DAP model on toy sine data and its dynamically changing decision boundaries.

Figure 2. Decision boundaries of toy ellipse data at {1st, 15th, 30th} target domains along the ellipse boundary. Note that data in the source domains and first 10 target domains are in same distribution. Each row contains results for an ellipse of different radius.

4.2. Rotated MNIST

We also evaluate our method on the MNIST handwritten digits dataset. The dataset contains 60,000 images. Each image is 28 x 28 pixels. We rotate these digits by a certain angle and split the dataset into a total of 20 domains divided between training and testing. We take images rotated between 0° and 15° to be our training source domains and images between 15° and 60° for target domains. Note that each image is seen at exactly one angle, so the training procedure cannot track a single image across different angles. Examples of the data from the Rotated MNIST dataset is shown in Figure 6.

4.3. Yearbook Portraits

We additionally study our method on the publicly available American High School Yearbooks dataset. The dataset contains 37,921 front-facing American high school yearbook photos over 120 years. Like the gradual domain adaption paper, we use the first 2000 images (1905-1935) as

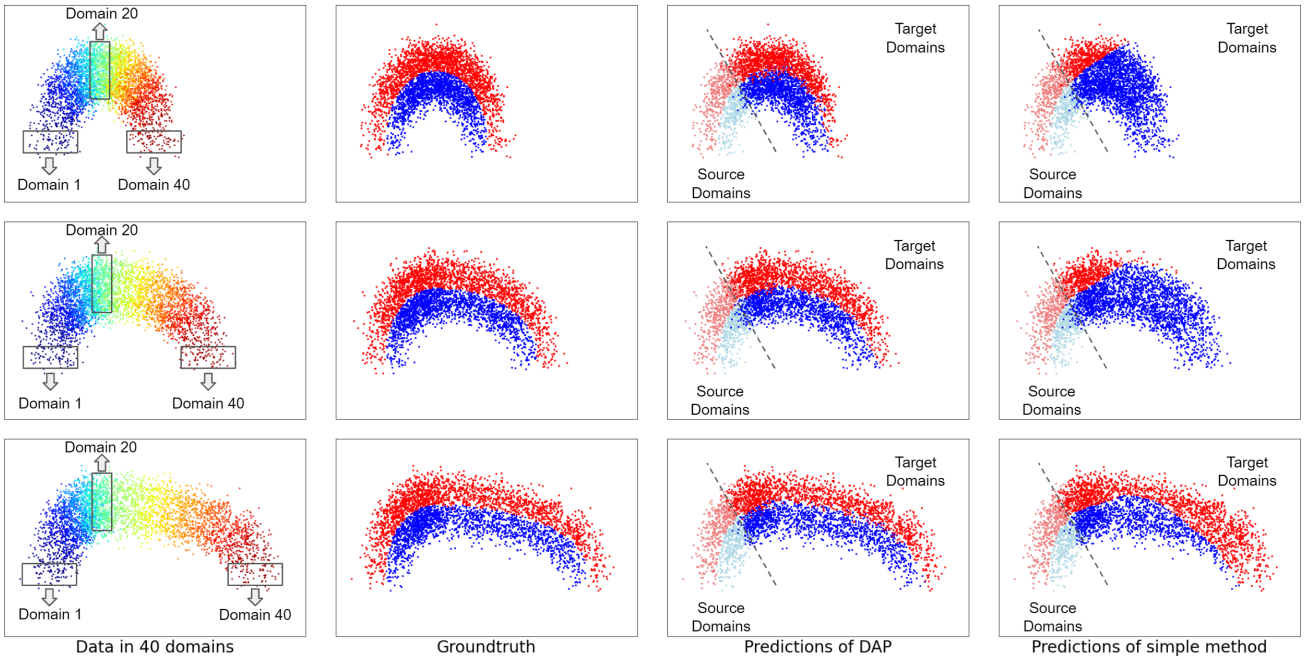

Figure 3. Results from ablation study on varying the radius of toy ellipse data. The radius values are {5, 10, 15} respectively for each row. The first column is the data. Blue is the source domain and orange is the target domain. The second column shows the ground truth classifications and decision boundary. The third column is the predicted decision boundary using our proposed method. The final column shows the results without using the predictive and stationary embedding vectors and simply training on the source domain. Source domains in third and fourth columns are marked with light colors while target domains are marked normally.

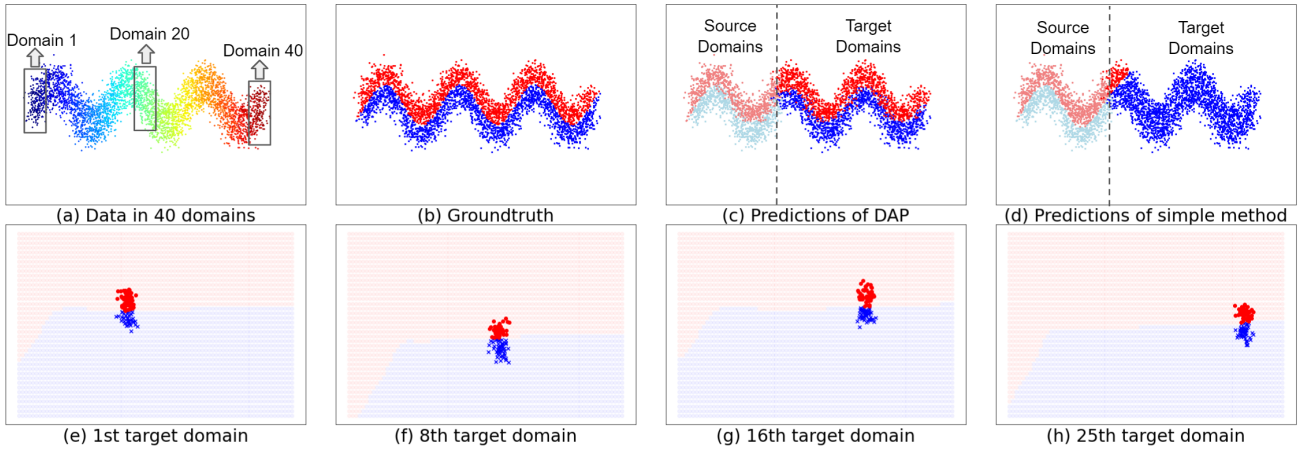

Figure 4. Results on the toy sine data. The first row shows the data, groundtruth and predictions of DAP and baseline models. The second row shows the decision boundary in several {1st, 8th, 16th, 25th} target domains.

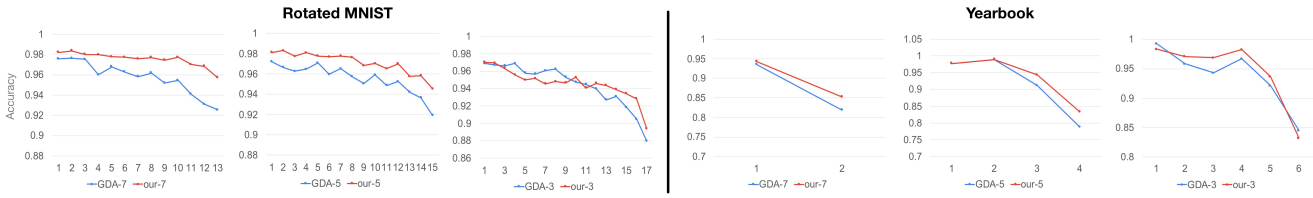

Figure 5. Comparison between Gradual Domain Adaptation(GDA) and our method when training includes different number of domains. The y-axis is the test accuracy in percent and the x-axis is index of the domain. The suffix of Our-X and GDA-X indicates number of domains used for the combined-source domain.

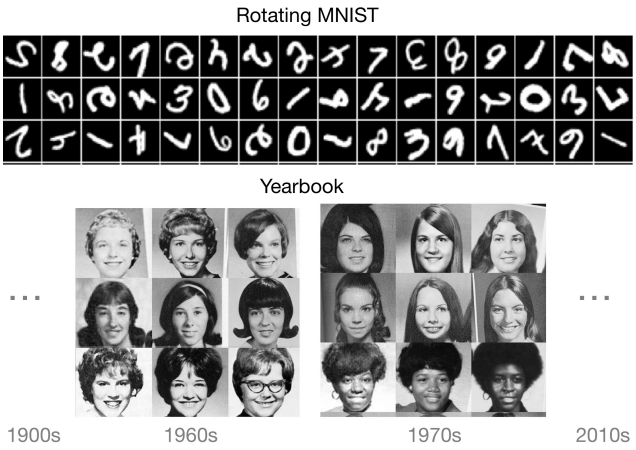

Figure 6. Examples of data from Yearbook[6] and Rotated MNIST.

the source, the next 14,000 (1935-1969) as the intermediate domains, and the next 2000 images as the target (1969 - 1973). The goal of our model is to classify the gender of the student. Examples of the data from the Yearbook Portraits dataset are shown in Figure 6.

5. Results

5.1. Gradual domain adaptation

We took Gradual Domain Adaptation(GDA) [12] as one of the baselines. We follow the dataset split introduced in section.4. We tested it on two datasets, Yearbook and Rotated MNIST. In our proposed method, we take at least two domains as the source domains so that the model can learn the "trending"; however, in GDA, it only needs one as source domain and then it can iteratively train (in an unsupervised way) on the following successive domains. So for a fair comparison, we group the same number of domains as we used for training our model into one source domain to test the GDA method. Figure.5 shows our method outperforms the baseline by a large margin, which further proves the efficacy of our method.

5.2. Ablation Studies

We conduct a series of ablation studies to understand the effects of several variables in our method such as the number of training domains and the batch size of each domain.

Intuitively, we expect that using a larger batch size from each domain for training will result in higher accuracy and similarly for using more training domains. We report in Table 1 the average accuracies over our test domains as well as the accuracy of the last test domain for the Rotated-MNIST dataset when varying the batch size of random sample from each domain. We consistently achieve over 96% test accuracy.

We also studied the effect of varying the number of domains used for training and find as expected that performance improves with more training domains. Recall there are a total of 20 domains for the Rotated-MNIST dataset. We record our results in Table 2. Surprisingly, there is a large improvement from using 3 training domain (0.9433) compared to just one training domain (0.6935). The trends we observe from our ablation agrees with our intuition that performance should increase with higher batch size and more training domains.

We also analyzed the decision boundaries of our model over time on the toy ellipse dataset. In Figure 2, we observe that our model is able to learn a decision boundary that assumes a general shape of an ellipse indicating that the model has learned to capture the underlying distribution of our data over time.

Batch size	Average test acc (%)	Last domain acc (%)
64	0.9627	0.9256
128	**0.9693**	0.9347
256	0.9679	**0.9397**

Table 1. Ablation over batch size

#	Average test acc (%)	Last domain acc (%)
1	0.6935	0.3223
3	0.9433	0.915
5	0.9711	0.9453
7	**0.9753**	**0.957**

Table 2. Ablation over the number of training domains

5.3. Future Work

For future work, we aim to conduct more extensive analysis of our methodology and implement more baselines to further evaluate the benefits of our approach. One paper we would like to investigate further is the Conditional Adversarial Domain Adaptation paper. They propose a framework that conditions the adversarial domain adaptation on discriminative information to enable alignment of multimodal distributions. Another relevant benchmark is the CyCADA paper which uses a cycle-consistency loss to adapt representations at both the pixel-level and feature-level. We will adapt these methods into our continuously indexed domain setup. Additionally, we'd like to investigate further into some of our current failure cases. Specifically, we find that our model is not able to fully capture the optimal decision boundary for the sine curve dataset which should be the tangent to the curve.

6. Conclusion

In this project, we investigate the problem of domain adaptation in a gradual shifting environment. We propose

to learn the shifting dynamics through drift-aware predictive coding and our work shows that DAP coding is informative and can be used to adapt decision boundaries to each domain and further improve the performance. We quantitative evaluate our approach against several datasets and we compare it against a strong state-of-the-art baseline, gradual domain adaptation. Future work can investigate different kinds of domain shifts and more experiments can be conducted on real dataset.

References

[1] Adeleh Bitarafan, Mahdieh Soleymani Baghshah, and Marzieh Gheisari. Incremental evolving domain adaptation. *IEEE Transactions on Knowledge and Data Engineering*, 28(8):2128–2141, 2016. 2

[2] Andreea Bobu, Eric Tzeng, Judy Hoffman, and Trevor Darrell. Adapting to continuously shifting domains. In *International Conference on Learning Representations Workshop*, 2018. 1

[3] Konstantinos Bousmalis, Nathan Silberman, David Dohan, Dumitru Erhan, and Dilip Krishnan. Unsupervised pixel-level domain adaptation with generative adversarial networks, 2017. 2

[4] Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan. Domain separation networks, 2016. 1, 2

[5] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario March, and Victor Lempitsky. Domain-adversarial training of neural networks. *Journal of Machine Learning Research*, 17(59):1–35, 2016. 3

[6] Shiry Ginosar, Kate Rakelly, Sarah Sachs, Brian Yin, and Alexei A Efros. A century of portraits: A visual historical record of american high school yearbooks. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 1–7, 2015. 6

[7] S. Ginosar, K. Rakelly, S. M. Sachs, B. Yin, C. Lee, P. Krähenbühl, and A. A. Efros. A century of portraits: A visual historical record of american high school yearbooks. *IEEE Transactions on Computational Imaging*, 3(3):421–431, 2017. 1, 4

[8] Judy Hoffman, Trevor Darrell, and Kate Saenko. Continuous manifold based adaptation for evolving visual domains. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 867–874, 2014. 2

[9] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei Efros, and Trevor Darrell. Cycada: Cycle-consistent adversarial domain adaptation. In *International conference on machine learning*, pages 1989–1998. PMLR, 2018. 1

[10] Guillermo Ortiz Jimenez, Mireille El Gheche, Effrosyni Simou, Hermina Petric Maretic, and Pascal Frossard. Cdot: Continuous domain adaptation using optimal transport. *arXiv preprint arXiv:1909.11448*, 2019. 2

[11] Guoliang Kang, Lu Jiang, Yi Yang, and Alexander G Hauptmann. Contrastive adaptation network for unsupervised domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4893–4902, 2019. 1

[12] Ananya Kumar, Tengyu Ma, and Percy Liang. Understanding self-training for gradual domain adaptation. *arXiv preprint arXiv:2002.11361*, 2020. 1, 2, 6

[13] Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I Jordan. Conditional adversarial domain adaptation. In *Advances in Neural Information Processing Systems*, pages 1640–1650, 2018. 1, 3

[14] Xingchao Peng, Yichen Li, and Kate Saenko. Domain2vec: Domain embedding for unsupervised domain adaptation. In *European conference on computer vision*, 2020. 2

[15] Rui Shu, Hung H Bui, Hirokazu Narui, and Stefano Ermon. A dirt-t approach to unsupervised domain adaptation. *arXiv preprint arXiv:1802.08735*, 2018. 1

[16] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation, 2017. 2

[17] Hao Wang, Hao He, and Dina Katabi. Continuously indexed domain adaptation. In *International Conference of Machine Learning*, 2020. 3

[18] Markus Wulfmeier, Alex Bewley, and Ingmar Posner. Incremental adversarial domain adaptation for continually changing environments. In *2018 IEEE International conference on robotics and automation (ICRA)*, pages 1–9. IEEE, 2018. 2

[19] Han Zhao, Remi Tachet des Combes, Kun Zhang, and Geoffrey J Gordon. On learning invariant representation for domain adaptation. *arXiv preprint arXiv:1901.09453*, 2019. 1