Adapting to Continuously Shifting Domains

Anonymous Authors[1]

Abstract

Domain adaptation typically focuses on adapting a model from a single source domain to a target domain. However, in practice, this paradigm of adapting from one source to one target is limiting, as different aspects of the real world such as illumination and weather conditions vary continuously and cannot be effectively captured by two static domains. Approaches that attempt to tackle this problem by adapting from a single source to many different target domains simultaneously are consistently unable to learn across all domain shifts. Instead, we propose an adaptation method that exploits the continuity between gradually varying domains by adapting in sequence from the source to the most similar target domain. By incrementally adapting while simultaneously efficiently regularizing against prior examples, we obtain a single strong model capable of recognition within all observed domains. Our method is applicable on a wide variety of learning settings, including visual classification and reinforcement learning in a video game domain.

1. Introduction

Imagine a self-driving car with a recognition system trained in mostly sunny weather conditions. Gradually, it starts to rain, and the self-driving agent must adapt to this change and continue to navigate the roads safely. We think of this weather change as a domain shift (Gretton et al., 2009) from a source domain, sunny weather, to a target domain, rainy weather. This domain shift phenomenon seriously affects the efficacy of the car's recognition model, since it was trained in sunny conditions and may not generalize well to the novelty of rain.

The typical supervised learning solution to this problem is to further fine-tune the recognition model on labeled datasets of the target, rainy, domain. However, these labels are often unavailable and it can be prohibitively difficult or expensive to obtain enough labeled data to properly fine-tune the large number of parameters employed by deep, multilayer networks. As such, we would like the network to adapt to the new domain in an unsupervised manner, without any need for labeled target data.

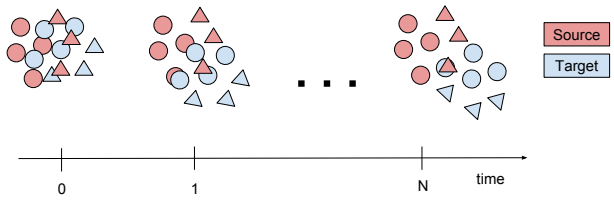


Figure 1. We consider the problem of adapting from a fixed source domain (denoted with red points) to a target domain which evolves over time (denoted with blue points). Our approach uses the continuity of the target domain shift over time to produce an adaptation method which learns to adapt to the current setting while producing a general model that efficiently remembers all prior settings.

Domain adaptation methods attempt to do just that: mitigate the harmful effects of domain shift by learning transformations that map the labeled source and the unlabeled target domains to a common embedding. This mapping is often achieved by optimizing the representation to minimize some measure of domain shift, such as maximum mean discrepancy (Tzeng et al., 2014; Long & Wang, 2015) or correlation distances (Sun & Saenko, 2016). More recently, adversarial approaches minimize the discrepancy between domains by training a generator to fool a discriminator by producing transformed source images that are indistinguishable from target images (Ganin et al., 2015; Tzeng et al., 2017).

Although these methods transfer well between similar domains, they produce poor results when the covariate shift is too large (Wulfmeier et al., 2017). This is precisely the case of sunny versus rainy weather in the earlier autonomous vehicle example. There, weather change is a gradual process that accumulates small shifts (e.g., darker and darker sky, incipient rain droplets) to produce large differences in domains over extended periods of time. We draw inspiration from this observation and posit that, in many scenarios, domains vary continuously and the shift cannot be effectively

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

captured in just two domains alone, as illustrated in Figure 1. Instead, we adapt iteratively from one source to many gradually shifted target domains by exploiting the continuity in the shift.

One issue that arises from this continuous adaptation procedure is a neural network's general tendency to forget past knowledge as it specializes to the current domain. This phenomenon of *catastrophic forgetting* (Ratcliff, 1990) happens in sequential training because the weights in the network that are important for previous domains are altered to adapt to the current domain. Our method corrects this issue by ensuring that at every adaptation stage the model continues to consistently classify previously seen examples. To enforce this constraint, we add a *replay loss* that forces previously recorded logits to match the current model's classification scores. Thus, a single model can perform continuous adaptation while maintain strong performance across all domains.

To summarize, in this work we tackle the problem of domain adaptation starting from one labeled source domain that continuously shifts into multiple successive unlabeled target domains. We show that it is important to uniquely adapt to different domains and present an algorithm that enables a single model to perform continuous adaptation in stages, from one domain to the next closest one, while consistently maintaining performance on all previously seen domains.

Over the next sections, we present our method for allowing a model to robustly adapt to continuously shifting domains while preserving high accuracy on previously seen data. In Section 2, we start from standard unsupervised adaptation models and explain the staging modifications needed to handle continuous shifts in multiple unlabeled target domains. Next, we introduce the notion of *replay*, which refers to holding on to the model's scores for a few examples in previous domains, and constraining the current stage model to match their scores. In Section 3, we present experiments that focus on visual classification for continuously rotated MNIST digits, and on video game play for Atari games that are gradually color inverted. In addition, we perform a hyperparameter study for different losses and supplemental structures that aid in remembering past knowledge.

2. Continuous Unsupervised Adaptation

In continuous adaptation, we are presented with a source domain S, and multiple target domains T_i that represent continuous shifts of S at time i. We assume access to source images X_s and labels Y_s drawn from a source domain distribution $p_s(x, y)$, as well as target images X_{t_i} drawn from target distributions $p_{t_i}(x, y)$, where there are no labeled observations. As such, we define $X_s : \{(x_1, y_1), ..., (x_N, y_N)|(x_i, y_i) \overset{iid}{\sim} p_s(x, y), \forall i\}$ and $X_{t_i} : \{x_1, ..., x_N|x_j \overset{iid}{\sim} p_{t_i}(x, y), \forall j\}$.

We additionally assume that the source domain is similar to the target domain at time t_0, that the target domain is smoothly varying, and that p_{t_0} is more similar to p_s than p_{t_1} is to p_s. In general, the target domain may change back to the source at some future time, e.g., full rotation. Our goal is to learn a single target representation M_t and classifier C_t that can correctly classify images from all target domains into one of K categories at test time, despite the lack of domain annotations. Since direct supervised learning on the target domains is not possible, continuous adaptation instead learns a source representation mapping, M_s, and a source classifier, C_s, and then adapts that model for use in the stream of target domains.

This paradigm poses a number of challenges. First, simply treating all target domains together as a single batch of targets ignores the continuity and fails to successfully learn to adapt to distributions that are farther away from the source (Wulfmeier et al., 2017). A continuous approach that adapts to every new target sequentially may also run into the problem of catastrophic forgetting: although classification on the current target domain is successful, performance on past domains is harmed because the network weights that are important for domains $T_{i-1}, T_{i-2}, ...$ are altered to fit into T_i's specifications (Wulfmeier et al., 2017). Prior work addresses this problem by storing a different model for each stage (Li & Hoiem, 2016; Rusu et al., 2016), but that quickly becomes unscalable as we progress through the sequence of domains.

We present a general framework for continuous adaptation with replay, where we evolve the model to the new distribution while simultaneously guiding it to not deviate too far from how it previously performed on the prior distributions. Figure 2 illustrates the structure of the proposed replay model. The cylinders represent the source and target domains. The model updates after every adaptation stage to account for another shifted target domain. After every stage i, we store the scores outputted by the adaptation model for a subset of the examples in target T_i. For subsequent stages $i + 1, i + 2, ...$, we add a replay loss to enforce high performance on the stored past examples. We substantially subsample every target domain response to allow scalability of our method over long periods. In this section, we discuss the processes of model staging, domain subsampling and matching, and replay loss selection.

2.1. Sequential Unsupervised Adaptation

We introduce an adaptation model that progressively evolves to correctly classify multiple shifted domains. Standard unsupervised adaptation effectively adapts between a single source distribution $p_s(x, y)$ and a single target distribution $p_t(x, y)$ by aligning features from both domains. In other words, they learn the source and target mappings, M_s and

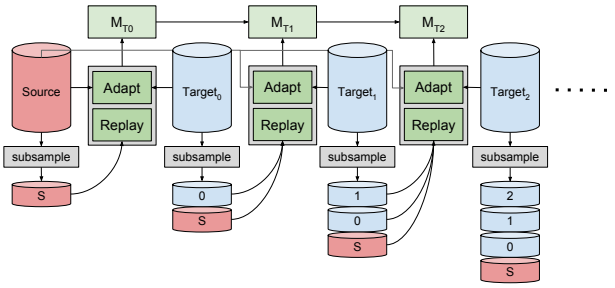


Figure 2. Proposed continuous replay model. At each stage, we save part of the adaptation predictions and use them as "soft" labels for the current domain. We enforce these past soft labels to be matched using a replay loss.

M_t, so as to minimize the distance between the empirical source and target mapping distributions:

$$M_t \leftarrow \arg\min_{M_t} d(M_s(X_s), M_t(X_t)). \tag{1}$$

Our method is general and any distance function d can be used. Common choices in recent works include the Kullback-Leibler divergence (Yang et al., 2012), Maximum Mean Discrepancy (MMD) (Gretton et al., 2008; Tzeng et al., 2014; Zhong et al., 2009), correlation alignment (Sun & Saenko, 2016), and adversarial loss (Liu & Tuzel, 2016; Tzeng et al., 2015; 2017; Ganin & Lempitsky, 2014).

When the distance between distributions is minimized, the source classification model, C_s, may be directly applied to the target representation as a target classifier C_t; we can, thus, denote both as C, and eliminate the need to learn a separate target classifier. We can now find M_s and C by optimizing the supervised objective:

$$M_s, C \leftarrow \arg\min_{M_s,C} \mathcal{L}_{cls}(C(M_s(X_s)), Y_s). \tag{2}$$

A common choice for $\mathcal{L}_{cls}$ is the cross-entropy loss, which results in the optimization:

$$\min_{M_s,C} \mathcal{L}_{cls}(C(M_s(X_s)), Y_s) = \\ - \mathbb{E}_{(x_s,y_s)\sim(X_s,Y_s)} \sum_{k=1}^{K} \mathbb{1}_{[k=y_s]} \log C(M_s(x_s)) \tag{3}$$

In the continuous problem statement, the goal is to minimize the distance between a single source and multiple targets:

$$M_t \leftarrow \arg\min_{M_t} d(M_s(X_s), M_t(\cup_{i=1}^{N} X_{t_i})). \tag{4}$$

The above mentioned domain alignment methods would simply conglomerate all target domains together and perform single source to single target adaptation. Unfortunately, standard unsupervised adaptation on a batch of target domains produces poor solutions for the posed optimization problem. Our first step towards improvement is adopting, instead, a sequential approach, where at every stage the model adapts to the next target domain. Starting from the labeled source domain S, we first guide the model to adapt to the unlabeled target domain T_1. Next, the source domain stays the same, but the target of interest becomes T_2. Since the domain continuity assumption dictates that T_1 is an intermediary from S to T_2, and the network has already adapted to T_1, the task of adapting from S to T_2 becomes much easier. This is due to the fact that, intuitively, by dividing the larger domain shift into smaller incremental shifts, the adaptation method has a smaller distance to minimize from domain to domain, which allows for more effective optimization solutions at every stage.

More generally, at each stage, T_i, we initialize the current target representation, M_{T_i}, using the adapted model from the previous stage, $M_{T_{i-1}}$. We then further adapt between the current target domain data, X_{T_i}, viewed under the current target model, and the source domain data, X_s, viewed under the original source model.

$$M \leftarrow M_{T_{i-1}} \tag{5}$$

$$M_{T_i} \leftarrow \arg\min_{M} d(M_s(X_s), M(X_{T_i})) \tag{6}$$

By continuing this process at every stage, we ensure successful adaptation to the next target domain. However, while staging alone enables models to more easily adapt, it does not solve the problem of catastrophic forgetting.

2.2. Continuous Replay Adaptation

We address the issue of forgetting previous domains by saving the scores for a few previously seen examples and introducing a *replay loss*, $\mathcal{L}_{replay}$, to enforce the response to be the same in the current stage model. This process is illustrated in Figure 2, where every stage's version of the adaptation model produces a mini-dataset with a few selected observations from their specific domain, together with the predicted classification scores. The subsampling is randomized at every stage and will be discussed in depth in Section 3.

Thus, M_t can be updated at every stage via a joint optimization of both the sequential unsupervised adaptation update together with the replay objective:

$$M_t \leftarrow \arg\min_{M_t} [d(M_s(X_s), M_t(X_{t_i})) \\ + \lambda \cdot \mathcal{L}_{replay}(C(M_t(X_p)), Y_p)] \tag{7}$$

Algorithm 1 CUA for continuous adaptation.

$M_s, C \leftarrow \arg\min_{M_s, C} \mathcal{L}_{cls}(C(M_s(X_s)), Y_s)$
$\{X_p, Y_p\} \leftarrow subsample(\{X_s, Y_s\}, \alpha)$
$M_t \leftarrow M_s$
for $i \in \{1...N\}$ **do**
  $M_t \leftarrow \arg\min_{M_t} d(M_s(X_s), M_t(X_{t_i}))$
  $\quad + \lambda \cdot \mathcal{L}_{replay}(C(M_t(X_p)), Y_p)$
  $\hat{Y}_{t_i} \leftarrow C(M_t(X_{t_i}))$
  $\{X_p, Y_p\} \leftarrow \{X_p, Y_p\} \cup subsample(\{X_{t_i}, \hat{Y}_{t_i}\}, \alpha)$
end for

where X_p and Y_p are the random samples and their predicted scores saved from previous domains, and λ is a replay weight that controls how much to optimize for past domain efficiency. When choosing the replay loss function, $\mathcal{L}_{replay}$, we experiment with both the above mentioned cross-entropy, and the standard mean squared error (MSE) loss:

$$\mathcal{L}_{replay}(C(M_t(X_p)), Y_p) = \frac{1}{N}\sum_{i=1}^{N}(C(M_t(X_p)) - Y_p)^2 \tag{8}$$

Algorithm 1 illustrates the described Continuous Unsupervised Adaptation (CUA) procedure, which sequentially adapts to an evolving target distribution while using replay of past examples to retain prior performance. The method begins by initializing a supervised source model using the labeled source data, and subsampling a few examples from the source data as replay data. A parameter α controls the subsampling rate by deciding how large of a fraction of the data to store. For every new target domain, we fit a new target representation M_t by adapting with distance metric d and replay loss $\mathcal{L}_{replay}$. Finally, we subsample α-rate data from the current target domain together with the predicted classification scores obtained under this stage's model.

3. Experiments

We now evaluate CUA for unsupervised classification adaptation to continuously shifting domains. We present two different adaptation scenarios, MNIST digit rotations and incremental color inversions for Atari. Surprisingly, both of these settings cause traditional unsupervised adaptation methods to fail when attempting to adapt to all variations together. In addition, we will show that for some of these shifts, the domain difference from the source to a particular target is large enough that traditional approaches even fail to adapt from source to that single target domain. We compare our model CUA against multiple state-of-the-art unsupervised adaptation methods that perform adaptation to a batch of target domains. In all of our experimental setups, our method significantly outperforms the competing approaches and approaches fully supervised performance.

3.1. MNIST rotations

The first continuous shift we consider is image rotations on MNIST digits. Our goal is to adapt from regular MNIST digits with rotation 0° to MNIST digits of various rotations. Figure 3 illustrates an example of the rotations in question. We designate rotation by 0° to be the labeled source domain, and rotations 45°, 90°, 135°, 180°, 225°, 270°, and 315° to be unlabeled target domains.

Implementation Details. The MNIST dataset contains 60000 training images of handwritten digits, and 10000 test images. The dataset has $k = 10$ classes, each corresponding to one digit. We randomly split the training set in half, assigning 30000 images to the source domain (rotation 0°). The remaining 30000 images are further split equally between the seven rotations which comprise the target domain variations. We preserve the marginal distributions over labels in each split. We use LeNet (Cun et al., 1990) as our base architecture in all experiments. As our unsupervised domain adaptation method to adapt between sequential domains we choose the recently proposed ADDA method (Tzeng et al., 2017).

Comparison Approaches. As our source model we train a supervised model on 0° and evaluate on the target domains with no adaptation. We also compare against recent unsupervised domain adaptation methods, DANN (Ganin et al., 2015) and ADDA (Tzeng et al., 2017) that we train on 0° MNIST and adapt to all target domains in batch.

Variants of CUA. As an ablation we consider multiple variants of our CUA model. First, we prove the importance of our replay objective (Section 2.2) when considering retention of information learned from prior adaptation shifts (see Figure 4. We also discuss the trade-off between remembering old domains and adapting to the current domain and demonstrate how the replay weight, λ, can be used to tune this trade-off. Finally, we will analyze the scalability of our method by demonstrating its ability to retain old information even with a very small sub-sampling rate, α.

3.1.1. Evaluation and Ablation of Replay

In Table 1, we compare the source only classification (no adaptation); the three unsupervised adaptation methods ADDA and DANN; CUA with no replay; our full CUA method; and the result of supervised training on all domains. All competing methods that do not use our framework fail catastrophically to adapt to the variety of target domains. Note, we used the original source code released with the UNIT (Liu et al., 2017) method, but were only able to achieve performance around 10%. Since this is far below source only performance we omit this result from our tables.

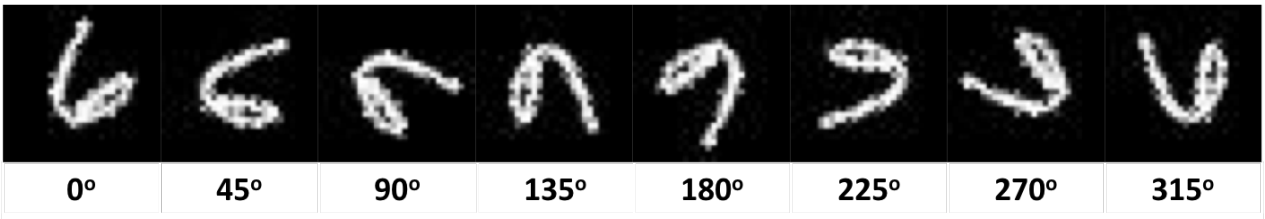


Figure 3. Every 45° rotation for an MNIST digit.

Method	**0°**	**45°**	**90°**	**135°**	**180°**	**225°**	**270°**	**315°**	**Average (%)**
Source	99.2	61.7	17.2	29.1	39.4	29.8	15.8	51.7	43.0 ± 0.8
ADDA	80.8	70.4	20.8	28.6	42.1	40.2	23.8	41.2	43.5 ± 1.2
DANN	98.6	64.7	19.9	28.4	41.4	32.9	24.2	67.3	47.2 ± 1.6
CUA - no replay (Ours)	51.6	15.1	32.7	38.7	30.4	27.1	73.6	96.0	45.7 ± 1.4
CUA (Ours)	**90.4**	**84.4**	**82.0**	**77.3**	**85.8**	**88.2**	**92.7**	**96.5**	**90.4 ± 1.6**
Target Supervised (Oracle)	96.9	96.7	96.8	97.4	96.6	96.5	96.8	96.4	97.0

Table 1. Rotated MNIST results for various adaptation methods. We evaluate each row on test data at rotations in 45° intervals. The last column contains the average over all test rotations.

The source model, DANN, and ADDA have high accuracy when tested on the source domain 0°, but fail to adapt to domains that are more distinct (i.e. 90° and larger rotations). CUA without replay is able to perform remarkably well on the current target domain, but fails when evaluated on past target domains, in other words suffers from catastrophic forgetting. Finally, our full method, CUA, clearly outperforms all other methods, with high accuracy both on the current and on past domains. On average, our method achieves 90.4% accuracy, a larger than 40% raw improvement over the next competing approach and nearing the performance of a fully supervised model.

A further comparison between CUA with and without replay reveals the dramatic impact that past data-matching has on maintaining high accuracy on past domains. Figure 4 shows that although both methods have comparable performance on current domains (blue lines), the replay loss dramatically helps against catastrophically forgetting previous domains (orange lines) and its impact is consistent across all domains (trend holds across rotations on x-axis).

For additional insight on what effect our staged replay framework has on classification, Figure 5 plots confusion matrices before adaptation, after batch adaptation, for CUA without replay and with replay, respectively. We present the confusion matrix for a test rotation of 135° after fully training all models. Examining the figure for the source-only baseline reveals that the domain shift is clearly very large, and, as a result, the network can only consistently classify the digit 0, which is already rotation-invariant. Batch and no replay CUA adaptations contribute minor performance improvements. With our method, the overall performance is quite striking, achieving an almost perfect classification score.

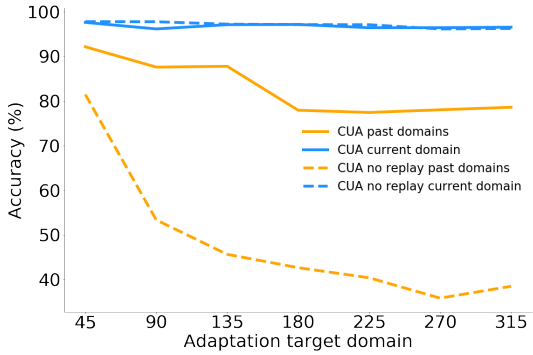
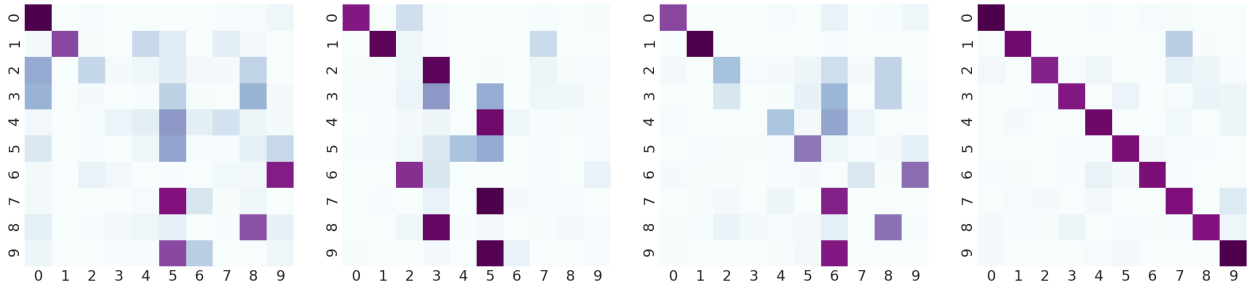


Figure 4. MNIST rotations accuracy reported as a function of the rotation amount. We compute the timelapse performance at each rotation evaluated as a past domain or at the time of adaptation to that domain (current domain). While the current domain accuracy is comparable for both our CUA and CUA no replay methods, past domain accuracy is dramatically higher and more consistent as the domain shifts when replay is included.

3.1.2. Trade-off: Adapting to New Domains versus Remembering Past Data

In this subsection we investigate the effect of varying the replay parameter λ that controls old data classification accuracy versus adaptation to the new domain. Figure 7 illustrates the expected behavior that, as λ increases, the current stage accuracy suffers, while the past stage one increases. The past stage accuracy plateau demonstrates that after the



(a) Source (b) Adapt Batch (c) CUA (no replay) (d) CUA (Full Model)

Figure 5. Confusion matrices for each model, evaluated on MNIST at the 135° orientation. Our method is able to correctly classify the vast majority of digits. In comparison, standard methods are either unable to effectively handle the domain shift or suffer from catastrophic forgetting, leading to degraded performance on previously seen domains.

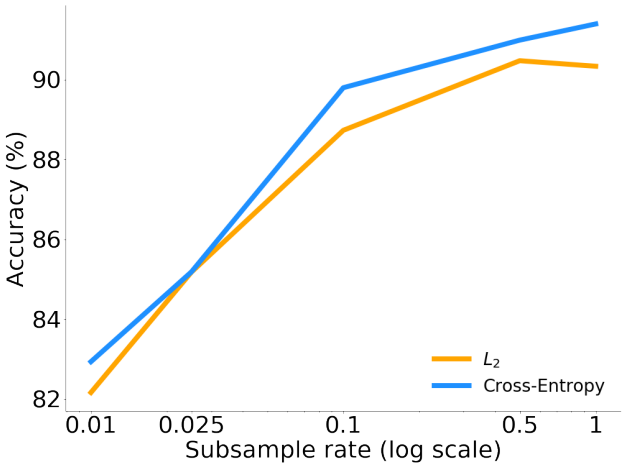
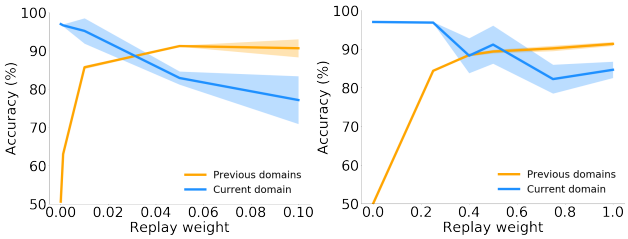


Figure 6. We report here the MNIST rotations accuracy averaged across all past domains together with the current domain. We study the the effect of the subsampling rate, α, on overall performance. α controls what fraction of the past data we store for replay. We experiment with a $\alpha \in \{0.01, 0.025, 0.1, 0.5, 1\}$ and find that CUA can handle extremely sparse cases with reasonable accuracy, demonstrating scalability.

network has paid enough attention to past examples, it cannot be further tuned to match old data better. We find that for the cross-entropy loss, values around $\lambda = 0.03$ are ideal, whereas for L_2, values around $\lambda = 0.4$ are more suitable.

3.1.3. Replay Loss Function

Our replay loss is agnostic to the particular objective used to enforce recall of old examples. As such, we evaluate two potential options here: the L_2 norm and cross-entropy between the prediction recorded during a prior stage and the current prediction. The cross-entropy calculation focuses on the one classification label that is most likely for a particular datapoint, and discards the rest of the signal for categories with lower classification scores. In theory, the L_2 norm pays



(a) Cross Entropy (b) L_2

Figure 7. We report here the MNIST rotations accuracy averaged across all past domains, orange, compared against the current domain, blue. To study of the effect of tuning the replay weight, λ, which controls the trade-off between remembering old examples and learning new examples, we plot performance vs λ with values from 0 to 1. We also experiment with two different replay losses for remembering the old data, Cross Entropy and L_2. In both cases, there is a setting of λ that produces strong adaptation performance while remembering and performing well on old data settings.

attention to the entire distribution of the scores, so intuitively we would expect it to match past data more accurately than the single signaled cross-entropy loss. However, in practice, we found that the method performed similarly regardless of which of these two losses we choose (see Figure 7).

3.1.4. Scalability Analysis

When adapting to continually shifting domains in the real world, scalability is a crucial component for allowing the model to evolve throughout many increasingly changing domains. We implement a subsampling rate, which is a parameter α that dictates how much past data to be saved for future replay. Intuitively, the fewer samples the framework stores, the less correctly the model will remember past domains.

Figure 6 shows average accuracy across all past domains and the current domain after the final rotation stage. This figure illustrates that as we subsample fewer examples, the accu-

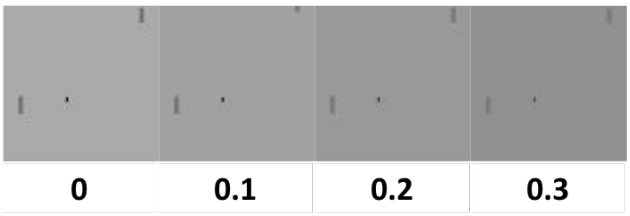


Figure 8. Example Pong frames for varying degrees of inversion. The frames have been resized to 84×84 and converted to grayscale, and are shown here as they are presented to the network.

racy decreases as expected, but not by a dramatic amount, suggesting that our method is highly scalable longterm. We plot five subsampling rates of $\alpha = 0.01, 0.025, 0.1, 0.5, 1$, and we see that a dramatic reduction of the saved data by 100 only loses fewer than 10 accuracy points as compared to storing the full dataset. This is, of course, significant, but still impressive given the very little past data the model replays.

3.2. Atari inversions

We further demonstrate the effectiveness of our method by evaluating it in a reinforcement learning setting—specifically, the task of learning to play Atari games. In the source domain, we assume a standard reinforcement learning setup, wherein at each timestep the agent in state s_t selects an action a_t from the set of legal game actions, $A = \{1, ..., K\}$. Upon taking action a_t, the environment transitions to some state s_{t+1} observed by the agent, and some reward r_t is obtained. The agent's goal is to maximize its cumulative reward.

We consider the task of adapting policies to a series of target domains. In each target domain, the agent can take actions and observe states. However, unlike in the source domain, no reward is available during training. Thus, standard reinforcement learning cannot be applied in this setting. Instead, we look to adapt the policy learned in the source domain according to the observations obtained in the target domain.

We emulate the Atari game Pong and choose a domain shift represented by color inversion. Surprisingly, even an incredibly simple color transformation breaks the performance of existing state-of-the-art models. We define color inversion as an operation parametrized by $\theta \in [0, 1]$, where every inverted pixel x_{inv} can be written as a function of the original x_{orig}:

$$x_{inv} = (1 - \theta) * x_{orig} + \theta * (1 - x_{orig}). \tag{9}$$

Figure 8 illustrates the gray-scale color change as θ varies for an Atari Pong frame. For $\theta = 0.0$, there is no inversion, while $\theta = 1.0$ would result in completely inverted frames.

	Inversion factor θ			
Method	0.0	0.1	0.2	0.3
Source only	21.0	21.0	17.6	−2.28
MMD (Long & Wang, 2015)	21.0	21.0	17.0	15.9
CUA (Ours)	21.0	21.0	21.0	21.0
Target with reward (Oracle)	21.0	21.0	21.0	21.0

Table 2. Adaptation across visual domains in an Atari setting. We train a base model for $\theta = 0.0$ using ACKTR (Wu et al., 2017), then adapt to the $\theta = 0.1, 0.2, 0.3$ environments without any reward, only observations. We report the reward obtained by the model averaged over 100 episodes. Despite the absence of rewards during training, which makes additional reinforcement learning infeasible, our method is able to recover full performance in the target domains. This provides further evidence for the robustness of our method.

For our experiments, the source domain has $\theta = 0.0$, i.e., unaltered, and the target domains have $\theta = 0.1, 0.2, 0.3$. Just as in the MNIST experiments, we compare our staged adaptation method against a source model and batch adaptation. The source model is an ACKTR baseline agent (Wu et al., 2017) trained for 50 million timesteps. The model consists of a network with 3 convolutional layers and 3 fully connected layers—the exact model definition can be found in the OpenAI baselines implementation (Dhariwal et al., 2017). We use this model, together with observations given by the emulator, as input to each adaptation method.

In this setting, we use another standard unsupervised domain adaptation approach of maximum mean discrepancy (MMD) (Tzeng et al., 2014; Long & Wang, 2015) as our distance function d:

$$d(M_s(X_s), M_t(X_t)) = ||\mathbb{E}[M_s(X_s)] - \mathbb{E}[M_t(X_t)]|| \tag{10}$$

In particular, we look to adapt the source and target models so that the output of their first fully connected layers are aligned with each other.

Table 2 shows the rewards obtained by the different methods on Pong. We report the average reward obtained over 100 episodes for various methods. The results show that our method is quite effective, recovering full performance across all target domains. Using a staged approach proves to be vital, as the model is unable to effectively adapt when presented with all target domains simultaneously. We also note that our method is robust, since despite being entirely unaware of the concept of reward, it is able to preserve the long-term dependencies necessary for performing well.

4. Related Work

Continuously changing domains pose significant challenges for robot learning and autonomous driving, since small incremental shifts cumulate to a large domain discrepancy overtime. The large domain shift between the training source domain and unlabeled target domain seriously affects the efficacy of machine learning models, as agents do not always have access to training data that is exactly representative of the intended testing scenario. In an attempt to solve this, many past methods have focused on creating feature transformations able to map domains to a space invariant to domain change (Lowry et al., 2016). Some other approaches predict the changes by synthesizing intermediate images between domains (Neubert et al., 2013) or retraining multiple experiences of the same visual scene (Churchill & Newman, 2013). However, neither method scales with an increase in target domains in the continuous shift settings.

Another approach, domain transfer learning, has studied both shallow (Gretton et al., 2009; Csurka, 2017) and deep methods (Tzeng et al., 2017; Sun & Saenko, 2016; Tzeng et al., 2014; Taigman et al., 2016; Liu et al., 2017). Recently, the domain adaptation community has been focusing on transferring deep neural network representations from a labeled source dataset to a target domain where labeled data is sparse or non-existent. The main strategy has been to learn representations by minimizing the difference between the source and target feature distributions (Gretton et al., 2009; Sun & Saenko, 2016; Tzeng et al., 2014). In Tzeng et. al.'s Adversarial Discriminative Domain Adaptation (ADDA) (Tzeng et al., 2017), the method guides feature learning by training a generator to fool the discriminator by producing images indistinguishable from target images, effectively minimizing discernibility between the source and target feature distributions. All these works treat the unsupervised domain adaptation problem as a batch transition without exploiting the continuity of the shifting domains, which significantly impacts their performance in the continuous problem setting.

Several other methods have attempted to tackle domain shift through domain generalization, which aggregates all information from multiple training domains or datasets to learn a shared invariant representation. Here, the focus is not on adapting the classifier to the target domain, because it is unknown. Instead, methods like (Motiian et al., 2017) learn at training time an embedding that maps to a domain invariant space. A related approach that uses domain randomization hypothesizes that with enough variability in training, the testing domain may appear to the model as just another variation (Tobin et al., 2017). The method attempts to make the model generalizable from simulations to real-world robots by introducing variability at training time in the form of randomized transformations to the original simulator. Unfortunately, both methods require a varied collection of labeled target domains and the knowledge that test domains will be relatively similar, which in our setting is unavailable. Furthermore, since the methods do not utilize the target domain information at all, it is unclear how well these models would adapt to domains of larger discrepancy than the ones in training.

A notable line of work is that of continuous manifold learning (Hoffman et al., 2014), where they adapt to evolving visual domains by learning a sequence of transformations on a fixed source representation. Another recent paper that is closest to our method discusses a similar incremental domain adaptation approach for continually changing environments (Wulfmeier et al., 2017). However, in both papers the authors are exclusively concerned with efficient adaptation for online streams of continuously shifted data, and do not focus at all on performance for past examples or the issue of catastrophic forgetting.

Other methods that tackle forgetting specifically, such as *progressive networks* (Rusu et al., 2016), which explicitly supports transfer across sequences of tasks in its architecture, or *elastic weight consolidation* (Kirkpatrick et al., 2016), which selectively slows down learning on the weights important for old tasks, focus exclusively on supervised transfer, which is unsuitable for continuous adaptation. Moreover, progressive networks increase in size as more data comes in, which quickly becomes not scalable in a prolonged continuous setting. "Learning without Forgetting" (Li & Hoiem, 2016) suffers from the same problem, because it involves saving all previous models and reclassifying data through them at every stage.

5. Conclusion

We have proposed a flexible framework for continuous unsupervised domain adaptation that enables single adaptation models to adapt to continual domain shifts while consistently maintaining performance on all domains. We show that even on small domain adaptation problems, such as continuously rotating MNIST digits and smoothly varying contrast in an Atari game, traditional adaptation methods fail catastrophically to cope with the evolving and diverse target domain. In, contrast our method is successfully able to recover near supervised learning performance on both the current domain as well as effectively and efficiently retain the information necessary to perform well on prior instantiations of the target domain.

References

Churchill, Winston and Newman, Paul. Experience-based navigation for long-term localisation. *The International Journal of Robotics Research*, 32(14):1645–1661, 2013.

doi: 10.1177/0278364913499193. URL https://doi.org/10.1177/0278364913499193.

Csurka, Gabriela. Domain adaptation for visual applications: A comprehensive survey. *CoRR*, abs/1702.05374, 2017. URL http://arxiv.org/abs/1702.05374.

Cun, Y. Le, Matan, O., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., Jacket, L. D., and Baird, H. S. Handwritten zip code recognition with multilayer networks. ii:35–40 vol.2, Jun 1990. doi: 10.1109/ICPR.1990.119325.

Dhariwal, Prafulla, Hesse, Christopher, Klimov, Oleg, Nichol, Alex, Plappert, Matthias, Radford, Alec, Schulman, John, Sidor, Szymon, and Wu, Yuhuai. Openai baselines. https://github.com/openai/baselines, 2017.

Ganin, Y. and Lempitsky, V. Unsupervised Domain Adaptation by Backpropagation. *ArXiv e-prints*, September 2014.

Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., and Lempitsky, V. Domain-Adversarial Training of Neural Networks. *ArXiv e-prints*, May 2015.

Gretton, A., Smola, AJ., Huang, J., Schmittfull, M., Borgwardt, KM., and Schölkopf, B. *Covariate shift and local learning by distribution matching*, pp. 131–160. MIT Press, Cambridge, MA, USA, 2009.

Gretton, Arthur, Borgwardt, Karsten M., Rasch, Malte J., Schölkopf, Bernhard, and Smola, Alexander J. A kernel method for the two-sample problem. *CoRR*, abs/0805.2368, 2008. URL http://arxiv.org/abs/0805.2368.

Hoffman, Judy, Darrell, Trevor, and Saenko, Kate. Continuous manifold based adaptation for evolving visual domains. pp. 867–874, 06 2014.

Kirkpatrick, James, Pascanu, Razvan, Rabinowitz, Neil C., Veness, Joel, Desjardins, Guillaume, Rusu, Andrei A., Milan, Kieran, Quan, John, Ramalho, Tiago, Grabska-Barwinska, Agnieszka, Hassabis, Demis, Clopath, Claudia, Kumaran, Dharshan, and Hadsell, Raia. Overcoming catastrophic forgetting in neural networks. *CoRR*, abs/1612.00796, 2016. URL http://arxiv.org/abs/1612.00796.

Li, Zhizhong and Hoiem, Derek. Learning without forgetting. *CoRR*, abs/1606.09282, 2016. URL http://arxiv.org/abs/1606.09282.

Liu, Ming-Yu and Tuzel, Oncel. Coupled generative adversarial networks. *CoRR*, abs/1606.07536, 2016. URL http://arxiv.org/abs/1606.07536.

Liu, Ming-Yu, Breuel, Thomas, and Kautz, Jan. Unsupervised image-to-image translation networks. *CoRR*, abs/1703.00848, 2017. URL http://arxiv.org/abs/1703.00848.

Long, Mingsheng and Wang, Jianmin. Learning transferable features with deep adaptation networks. *CoRR*, abs/1502.02791, 2015. URL http://arxiv.org/abs/1502.02791.

Lowry, Stephanie, Sünderhauf, Niko, Newman, Paul, Leonard, John J., Cox, David, Corke, Peter, and Milford, Michael J. Visual place recognition: A survey. *IEEE Transactions on Robotics*, 32(1):1–19, 2016. doi: 10.1109/TRO.2015.2496823. URL https://eprints.qut.edu.au/105651/.

Motiian, Saeid, Piccirilli, Marco, Adjeroh, Donald A., and Doretto, Gianfranco. Unified deep supervised domain adaptation and generalization. *CoRR*, abs/1709.10190, 2017. URL http://arxiv.org/abs/1709.10190.

Neubert, P., Snderhauf, N., and Protzel, P. Appearance change prediction for long-term navigation across seasons. In *2013 European Conference on Mobile Robots*, pp. 198–203, Sept 2013. doi: 10.1109/ECMR.2013.6698842.

Ratcliff, Roger. Connectionist models of recognition memory: constraints imposed by learning and forgetting functions. *Psychological review*, 97 2:285–308, 1990.

Rusu, Andrei A., Rabinowitz, Neil C., Desjardins, Guillaume, Soyer, Hubert, Kirkpatrick, James, Kavukcuoglu, Koray, Pascanu, Razvan, and Hadsell, Raia. Progressive neural networks. *CoRR*, abs/1606.04671, 2016. URL http://arxiv.org/abs/1606.04671.

Sun, Baochen and Saenko, Kate. Deep CORAL: correlation alignment for deep domain adaptation. *CoRR*, abs/1607.01719, 2016. URL http://arxiv.org/abs/1607.01719.

Taigman, Yaniv, Polyak, Adam, and Wolf, Lior. Unsupervised cross-domain image generation. *CoRR*, abs/1611.02200, 2016. URL http://arxiv.org/abs/1611.02200.

Tobin, Joshua, Fong, Rachel, Ray, Alex, Schneider, Jonas, Zaremba, Wojciech, and Abbeel, Pieter. Domain randomization for transferring deep neural networks from simulation to the real world. *CoRR*, abs/1703.06907, 2017. URL http://arxiv.org/abs/1703.06907.

Tzeng, Eric, Hoffman, Judy, Zhang, Ning, Saenko, Kate, and Darrell, Trevor. Deep domain confusion: Maximizing for domain invariance. *CoRR*, abs/1412.3474, 2014. URL http://arxiv.org/abs/1412.3474.

Tzeng, Eric, Hoffman, Judy, Darrell, Trevor, and Saenko, Kate. Simultaneous deep transfer across domains and tasks. *CoRR*, abs/1510.02192, 2015. URL http://arxiv.org/abs/1510.02192.

Tzeng, Eric, Hoffman, Judy, Saenko, Kate, and Darrell, Trevor. Adversarial discriminative domain adaptation. *CoRR*, abs/1702.05464, 2017. URL http://arxiv.org/abs/1702.05464.

Wu, Yuhuai, Mansimov, Elman, Grosse, Roger B, Liao, Shun, and Ba, Jimmy. Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation. In *Advances in Neural Information Processing Systems 30*. 2017.

Wulfmeier, M., Bewley, A., and Posner, I. Incremental Adversarial Domain Adaptation. *ArXiv e-prints*, December 2017.

Wulfmeier, Markus, Bewley, Alex, and Posner, Ingmar. Addressing appearance change in outdoor robotics with adversarial domain adaptation. *CoRR*, abs/1703.01461, 2017. URL http://arxiv.org/abs/1703.01461.

Yang, Pei, Gao, Wei, Tan, Qi, and Wong, Kam-Fai. Information-theoretic multi-view domain adaptation, 07 2012.

Zhong, Erheng, Fan, Wei, Peng, Jing, Zhang, Kun, Ren, Jiangtao, Turaga, Deepak, and Verscheure, Olivier. Cross domain distribution adaptation via kernel mapping. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, pp. 1027–1036, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-495-9. doi: 10.1145/1557019.1557130. URL http://doi.acm.org/10.1145/1557019.1557130.