LEARNING TO ADAPT TO SEMANTIC SHIFT

Anonymous authors
Paper under double-blind review

ABSTRACT

Machine learning systems are typically trained and tested on the same distribution of data. However, in the real world, models and agents must adapt to data distributions that change over time. Previous work in computer vision has proposed using image corruptions to model this change. In contrast, we propose studying models under a setting more similar to what an agent might encounter in the real world. In this setting, models must adapt online without labels to a test distribution that changes in semantics. We define two types of semantic distribution shift, one or both of which can occur: *static shift*, where the test set contains labels unseen at train time, and *continual shift*, where the distribution of labels changes throughout the test phase. Using a dataset that contains both class and attribute labels for image instances, we generate shifts by changing the joint distribution of class and attribute labels. We compare to previously proposed methods for distribution adaptation that optimize a fixed self-supervised criterion at test time or a meta-learning criterion at train time. Surprisingly, these provide little improvement in this more difficult setting, with some even underperforming a static model that does not change parameters at test time. In this setting, we introduce two models that "learn to adapt"—via recurrence and learned Hebbian update rules. These models outperform both previous work and static models under both *static* and *continual* semantic shifts, suggesting that "learning to adapt" is a useful capability for models and agents in a changing world.

1 INTRODUCTION

A traditional assumption in machine learning is that models are trained and tested on the same distribution of data. However, it is often the case that the test distribution may differ from the training distribution and can also change significantly over time. Examples include: evolving road and weather conditions for self-driving cars, new rooms and scenes for indoor robots, and cultural shifts over time for online posts. Unfortunately, model performance tends to degrade as the data distribution diverges from the training distribution, since models remain unchanged once deployed (Torralba & Efros, 2011). Since supervised learning is relatively weak to distribution shifts, current mitigation approaches require repeatedly re-collecting labels and retraining models. This can be expensive and impractical; we would prefer models that adapt online to new data in an unsupervised fashion. In this paper, we investigate methods for adapting models to changing data distributions, focusing on image classification.

Prior work proposes multiple settings that address different aspects of the problem. In the unsupervised domain adaptation (UDA) setting (Vapnik, 2006; Quionero-Candela et al., 2009; Joachims, 1999; Ganin & Lempitsky, 2015), access to the full test distribution at training time is typically assumed; this is unrealistic for models deployed in the real world. More complex forms of UDA proceed through a sequence of separate discrete domains in the test set (Lao et al., 2020; Wu et al., 2019) (e.g. images $\rightarrow$ paintings $\rightarrow$ sketches) or shift the test distribution continuously from a start point to an end point (Kumar et al., 2020; Wulfmeier et al., 2018; Liu et al., 2020a) (e.g., images of cars from 1950 $\rightarrow$ cars from 2010). These don't model the fact that real world distribution shifts can be discrete, continuous, cyclic patterns (e.g., day-night cycles or seasons), or a combination of all three.

We first present a realistic test setting, *Semantic Domain Shift*, that aims to address the above issues. In this setting:

- The test set may contain labels unseen at training time. We term this *static shift*.
- The distribution of labels may change throughout the test phase. We term this *continual shift*.
- After training, models must adapt *incrementally* during the test phase without access to gold-standard labels.

While a number of non-neural approaches (Hoffman et al., 2014; Kulis et al., 2011; Gong et al., 2012; Gopalan et al., 2011; Saenko et al., 2010; Fernando et al., 2013; Jain & Learned-Miller, 2011) for incrementally adapting to domain shift have been proposed, little work has addressed adaptation for deep neural networks. One exception is Test-Time-Training (Sun et al., 2020), which optimizes a self-supervised loss at training and testing time. Additionally, TENT (Wang et al., 2021) updates normalization parameters by minimizing prediction entropy at test time. However, these works only adapt to synthetic distribution shifts (applying image corruptions including Gaussian blur, salt and pepper noise) as opposed to natural ones (e.g., weather, lighting, style, attributes). Furthermore, by requiring training at test time, they impose extra computational requirements and add latency. This may not be possible in resource-constrained settings. Finally, Zhang et al. (2020) introduce *Adaptive Risk Minimization* (ARM), a metalearning objective for test sets that contain different subgroups than the training set, but do not consider shifting label distributions.

We introduce here a class of models that directly "Learn to Adapt" to *semantic domain shift*. We evaluate two classes of models: learned Hebbian plasticity and recurrent models. The earliest proposed mechanism for unsupervised learning and adaptation in neural networks is *Hebbian plasticity* (Hebb, 1949). By modifying synaptic weights between an input and output neuron according to the correlation of their activities, Hebbian plasticity is believed to play an important role for adaptation and learning in animals (Morgan; Sardi et al., 2020; Noguchi et al., 2004). Given the biological plausibility, we test a learned version of Hebbian Plasticity, building off Najarro & Risi (2020) who use it to solve reinforcement learning problems. Recurrent networks are also thought to play a role in biological adaptation (Kohn, 2007). They have been used in learning-to-learn (Andrychowicz et al., 2016; Schmidhuber, 1992) and for their ability to remember information over time (Haviv et al., 2019). We test a multi-layer LSTM (Hochreiter & Schmidhuber, 1997), training both with i.i.d. sampling and under *continual shift* at train time.

Using the MIT-States dataset (Isola et al., 2015), which labels each image instance with both a *class* annotation and an *attribute* annotation, we construct a test setting for evaluating model performance on *static shift*, *continual shift*, and both together. Surprisingly, we found that TENT, TTT, and ARM did not improve performance significantly in our setting, sometimes underperforming a single static model; cf. Section 4.3. We demonstrate that models trained directly for adaptation can outperform these methods without requiring training at test time.

Our Contributions. We introduce a more realistic test setting where domain shifts occur *semantically*. We show that current methods for adaptation based on optimizing a fixed self-supervised criterion perform comparably to a model that does not change its weights at test time, sometimes even underperforming. We demonstrate that both a recurrent model and learned hebbian update rules outperform previous methods that optimize a fixed self-supervised criterion at test time or optimize a meta-learning criterion at train time.

2 Related Work

2.1 Settings

Domain Adaptation. Many variants of Unsupervised Domain Adaptation (Vapnik, 2006), each focusing on different aspects of the problem have been explored. Generally, the majority of settings (Zhao et al., 2020) assume access to the entire test set at training time. Compound domain adaptation (Liu et al., 2020b) contains multiple domains with real-world variations (e.g., lighting, weather, etc.), but also assumes access to the full test distribution at train time. Open set DA (Busto & Gall, 2017; Busto et al., 2020) is similar to our setting in that novel labels can appear at test time. However, the setting tasks the model with predicting whether an instance has a novel label or not. In our setting, the model is tasked with explicitly predicting the label. Other settings (Kumar et al., 2020; Wulfmeier et al., 2018; Liu et al., 2020a) have proposed adapting to a gradually changing distribution,

but they assume that the test distribution shifts continuously from a start distribution (often the train distribution) to an end distribution, which is a limiting assumption for the real world (e.g. day-night cycles or seasons). Wang et al. (2020) propose a setting where the test distribution continually shifts along multiple axes (e.g. digit rotation and class), but they are only able to evaluate on simple images. Other settings focus on subpopulation shift Santurkar et al. (2021), in conjunction with domain generalization Koh et al. (2021), but do not have a shift in the predicted label distribution across train and test. Koh et al. (2021). Wu et al. (2019) propose a realistic setting most similar to ours, with 2 minor but important differences: We primarily focus on predicting the shifting labels in an image classification task, where they focus on remaining invariant to the changing test distribution for the task of semantic segmentation.

Continual Learning. Both the ability to adapt continually and the ability to continually learn are crucial for models and agents operating in the real world. In the continual learning setting (Parisi et al., 2019), a model must sequentially learn to solve a series of tasks without *catastrophic forgetting* - forgetting how to solve earlier tasks it was trained on (Nguyen et al., 2019). Our setting can be straightforwardly compared to the continual learning setting, with one subtle, but important difference.

The continual learning setting consists of two phases. First, the *training phase*: where models are incrementally provided tasks with *labeled* instances. Second, the *testing phase*: where frozen models are tested on *unseen unlabeled* instances for those tasks. Our setting also consists of two phases. First, the *training phase*, where models are provided a training set of *labeled* instances they can arbitrarily use for learning. Second, the *testing phase*: where models are asked to make predictions on *unseen unlabeled* instances under a series of changing tasks or conditions. Generally, Continual Learning asks a model to *learn* from a continually changing distribution and not *forget*, whereas our setting asks models to *learn* from a fixed distribution and *adapt* to a continually changing test distribution.

van de Ven & Tolias (2019) split experimental setups for continual learning into 3 categories: Task-incremental learning, Domain-incremental learning, and Class-incremental learning. In task-incremental learning, models are provided with the task identity. In our case, the test-time analogue would correspond to asking the model: "With a given possible set of attributes, what is the class of this instance?" or "With a given possible set of classes, what is the attribute of this instance?". In domain-incremental learning, models must solve the task at hand without being provided with the task identity. In our case, the test-time analogue would correspond to asking the model: "Without knowing the attribute, what is the class of this instance?" or "Without knowing the class of this instance, what is its attribute?". In class-incremental learning, models must solve the task at hand and predict the task identity. In our case, the test-time analogue would correspond to asking the model: "What are the attribute and class of this instance?".

We adopt the class-incremental structure and procedure from continual learning for constructing an ordering over examples at test time (See Sec 3). We evaluate the impact of catastrophic forgetting via repeating label subsets at different times during the test phase.

Setting	Static Shift	Continuous Shift (class shift)	Continuous Shift (class & attribute shift)
Compare with:	Domain Adaptation	Class-Incremental Learning	Class-Incremental Learning
Difference:	Shift in label distribution.	Labels not given	Labels not given, more complex semantic shift.

Table 1: Comparing the components of our setting with existing settings.

2.2 METHODS

Self-Supervised Adaptation. Previous work has proposed adapting to a (potentially changing) test-distribution by minimizing a self-supervised criterion on instances throughout the test phase. TENT (Wang et al., 2021) updates model normalization parameters (batch-normalization scale and shift (Ioffe & Szegedy, 2015)) by minimizing the mean entropy of model predictions on each test batch.

Test-Time-Training (Sun et al., 2020) adds a linear self-supervised head to the feature backbone with a rotation prediction (Gidaris et al., 2018) proxy task at training time. At test time, they update the parameters of the convolutional backbone to minimize the same rotation prediction loss on each test batch or instance. TENT and TTT shift the test distribution by applying image corruptions including, but not limited to: Gaussian blur, salt and pepper noise. In contrast, we change the distribution of labels (i.e. semantics) encountered at test time. In the interest of experimental completeness, we also evaluate our methods on the image corruption task (See the appendix). TENT builds on work by Li et al. (2016), who proposed adapting by using running statistics at test time to compute mean and standard deviation for batch-normalization layers.

Learning synaptic update rules. In contrast to artificial neural networks, biological neural networks learn rapidly, display significant adaptive behavior, and adjust to new domains quickly (Morgan; Sardi et al., 2020; Noguchi et al., 2004). Synaptic plasticity is believed to play an important role in adaptation and learning, particularly through local mechanisms that rely solely on local neuron-specific activity to modulate synaptic connections. The most well-known and earliest proposed mechanism, *Hebbian plasticity*, (Hebb, 1949) states: "Neurons that fire together, wire together" – concretely that the weight of a synapse betweeen neurons changes proportionally to correlated activity between the two neurons. Previous work has used Hebbian plasticity for continual learning (Thangarasa et al., 2020) on visual tasks, as well as learning-to-learn (Miconi, 2016), where they demonstrate capabilities for pattern completion, one-shot learning, and reversal learning. Other work has used differentiable plasticity to outperform standard LSTMs on language modeling tasks with an order of magnitude fewer parameters (Miconi et al., 2019). Closest to our approach is Najarro & Risi (2020), where connection-specific Hebbian update rules are learned to perform selected reinforcement learning tasks. We build off their formulation, adding a momentum term to ease optimization and weight normalization to deal with exploding weights.

3 Semantic Domain Shift

We assume a dataset where each image has two annotations: attributes (e.g., "wet," "melted," "deflated") and classes (e.g., "dog," "chocolate," "ball"). This allows us to predict unseen compound labels during testing without requiring test-time supervision. Additionally this allows us to examine more complex semantic shifts between the train and test set than a difference in relative frequencies of classes. For example, a model can be asked to predict the class of a wet dog when it has only seen dry dogs (and wet non-dog objects) at train time.

3.1 Static Shift

We define *static shift* as a shift in the distribution of labels between the train and test set. The simplest form of static shift is a shift in relative frequencies of classes between the train and test sets (e.g., the test set consists almost entirely of dogs when very few dogs are seen at train time). We use the fact that our assumed dataset contains both class and attribute labels to evaluate more complex *static shift*. We build a test set consisting of class-attribute combinations that are *unseen* at train time while all attributes and classes are *seen* at train time. In order to do this, we iteratively randomly sample compound labels with at least 5 instances that are not the only label containing a given class or attribute in the training set, moving those instances into the test set until it has at least 20% of the examples in the dataset. Once the dataset has been split into a train-val set and a test set, we conduct the same procedure on the train-val set to generate the training set and a validation set (with at least 10% of the examples in the train-val set).

3.2 Continual Shift

We define *continual shift* as a setting where the distribution of labels changes throughout the test phase. For example, early in the test phase, a model may see mostly dogs and cats, midway through mostly cats and trees, and late in the test set see dogs again.

Class Shift. We adopt the task-based ordering used in continual learning, where the test set is split into a number of tasks such that each class appears only within a single task. These tasks may occur

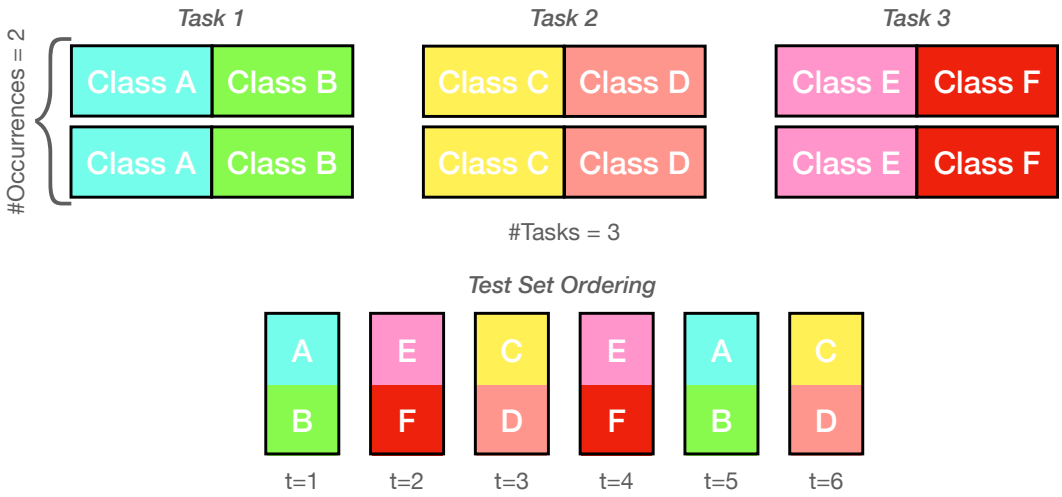

Figure 1: Classes are split into tasks, and each task is split into occurrences/subsets such that all classes in the task appear in each subset. We then randomly order these subsets and the contained instances to generate an ordering for the test set.

several times over the test phase; every example is still seen once. In order to construct an ordering for the test set, we use the following procedure.

Classes are randomly divided equally into k tasks. We split each task into n subsets or occurrences. This is done by splitting each class in a task into n subsets, and joining these subsets across classes in a task. This results in n subsets for each task, each of which contains examples from all classes in the task. Given k tasks, each split into n subsets, we randomly order the examples within each subset, then randomly order the $k * n$ subsets to produce a test set ordering; see Figure 1.

The division of classes into k tasks ensures that the distribution of labels changes over the test-set ordering. The division of each task into n occurrences allows us to test the effects of catastrophic forgetting and repetition on model adaptation. For example, with $n = 1$, once all the instances in a given task have been evaluated, no instances from the same classes will be seen for the rest of the test set. As n grows larger, the test set ordering approaches i.i.d. sampling.

Class-Attribute Shift. In this setting, we modify *class shift* to make it more difficult. When we split each class into n subsets, we do so in an attribute-disjoint way, so that no examples of the same class share the same attribute across the subsets. Since intraclass visual differences can often be larger than interclass visual differences due to attributes (e.g., there exists a larger visual difference between "Scrambled Eggs" and "Hardboiled Eggs" than between "Hardboiled Eggs" and "Ping Pong Balls"), it can be difficult to classify an object instance where the instance's class has not been seen with that attribute before. In the *Class Shift* setting, a model could potentially use temporal context (e.g., scrambled egg images appearing close in time together with hardboiled egg images) to determine the label. This becomes more difficult in the *Class-Attribute Shift* setting, particularly as the number of occurrences n increases.

4 Experiments

4.1 Experimental Setup

Dataset. We use MIT-States (Isola et al., 2015), a dataset of objects, scenes, and materials in transformed states. The dataset consists of $63,440$ images depicting 245 nouns (classes) modified by a total of 115 adjectives (attributes). Concretely, each example consists of an image, along with 2 labels: the *class* and an *attribute* or state for the object. The dataset was originally collected to study transformations in the physical world carrying semantic meaning (e.g. wetting, bending, aging). Images were collected of objects in transformed states, with transformations common across multiple object classes.

Model Architectures & Loss & Training All models use a Resnet-18 backbone (He et al., 2015), with two independent prediction heads. Adaptive models initialize backbone parameters from a resnet-18 pretrained on Imagenet (Russakovsky et al., 2015). We train all models with a sum of cross-entropy losses on attribute prediction and class prediction. Detailed descriptions of hyperparameters as well as the setup for fine-tuning models under class-shift are included in the appendix (**??**).

4.2 METHODS EVALUATED

We evaluate both *static* models that do not change parameters or state at test time, and *adaptive* models which update their parameters at test time. We compare to previous work that update parameters: BN (Li et al., 2016), TENT (Wang et al., 2021), and Test-Time-Training (Sun et al., 2020), giving a high level description below. We point the reader to Zhang et al. (2020) and Kang et al. (2020) for descriptions of the Adaptive Risk Minimization and τ-norm methods.

Static Baseline. In order to predict both classes and attributes, we combine a convolutional backbone f with two independent MLP heads g_c and g_a for predicting the class and attribute, respectively. The loss for a given instance x with class $\hat{y}_c$ and attribute $\hat{y}_a$ is then:

$$\mathcal{L}_{\text{sup}} = \mathcal{L}_{\text{CE}}(\hat{y}_c, g_c(f(\boldsymbol{x}))) + \mathcal{L}_{\text{CE}}(\hat{y}_a, g_a(f(\boldsymbol{x}))) \,, \tag{1}$$

where $\mathcal{L}_{\text{CE}}$ denotes the cross-entropy loss. We use a Resnet-18 (He et al., 2015) as a backbone for all methods. The MLP head consists of a linear layer that doubles the feature size (from 512 to 1024), a ReLU nonlinearity (Nair & Hinton, 2010), and a final linear layer that outputs class scores. Further methods build on this architecture.

Adaptation by Batch Normalization. Li et al. (2016) propose using test-time batch statistics to compute the mean and standard deviation for batch normalization layers instead of using values obtained from the training set. This can be easily accomplished by setting the batch normalization layers in the network into "training mode."

TENT. Wang et al. (2021) propose adapting during testing by minimizing the entropy of model predictions to update batch normalization scale and shift parameters. After every test batch, we minimize the sum of entropies for attribute and class predictions, updating the batch normalization scale and shift parameters. They also set batch normalization layers to use running statistics at test time, as in Li et al. (2016).

$$\mathcal{L}_{\text{TENT}} = H(g_c(f(\boldsymbol{x}))) + H(g_a(f(\boldsymbol{x}))) \,, \tag{2}$$

Test-Time-Training (TTT). Sun et al. (2020) propose adding a linear self-supervised head g_{ss} and a self-supervised loss $\mathcal{L}_{ss}$. Following Sun et al. (2020), we use rotation prediction as the self-supervised task, resulting in training loss:

$$\mathcal{L}_{\text{TTT}} = \mathcal{L}_{\text{sup}} + \mathcal{L}_{\text{CE}}(g_{ss}(\text{r}, f(\text{rotate}(\boldsymbol{x}, \text{r})))) \,, \tag{3}$$

where the rotation r is randomly selected from $[0, 90, 180, 270]$ degrees. At test time, TTT updates parameters for the backbone f by minimizing the rotation prediction loss.

LSTM: Memory via Recurrence. We seek to learn a recurrent mechanism for modulating image features given the history of images so far. The model proceeds through a given batch of examples sequentially (feature extraction by the backbone can be done in parallel). We insert a 2 layer residual LSTM with a hidden size of 512 (same as the resnet backbone feature size), between the model backbone and the predictor heads. We optimize using BPTT over the batch, detaching gradients for the LSTM state between batches. The LSTM state is reset to zero every epoch.

A Hebbian Learning Rule. We seek to learn connection-specific Hebbian update rules that will convert a randomly initialized layer into one with high performance on the task. We replace the first linear layer in each prediction head with a Hebbian layer (defined below), leaving the rest of the network architecture unchanged. The model proceeds through a given batch of examples sequentially (feature extraction by the backbone can be done in parallel). We build off work by Najarro & Risi (2020) in optimizing Hebbian rules for reinforcement learning, using a modified generalized ABCD rule (Soltoggio et al., 2007) for updating weights

Concretely, at time t, A Hebbian layer with weights $\boldsymbol{W}^t$ calculates the *pre-update output* $\boldsymbol{o}^t = \boldsymbol{W}^t \boldsymbol{x}^t$ for an input $\boldsymbol{x}^t$. The weights are then updated with a 4 factor rule combined with a momentum term $\Delta W_{i,j}^{t-1}$. The 4 factor rule is a linear combination of 4 terms: the pre-synaptic post-synaptic correlation $x_i^t o_j^t$, the pre-synaptic activation $\boldsymbol{x}_i$, the post-synaptic activation o_j and the current weight $W_{i,j}^t$:

$$\Delta W_{i,j}^t = \eta_{i,j} \cdot (A_{i,j} \cdot x_i^t o_j^t + B_{i,j} \cdot x_i^t + C_{i,j} \cdot o_j^t - D_{i,j} \cdot W_{i,j}^t) + E_{i,j} \cdot \Delta W_{i,j}^{t-1} \,, \tag{4}$$

where i and j index input and output neurons, respectively, and $\boldsymbol{\eta}$, $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$, $\boldsymbol{D}$, $\boldsymbol{E}$ are connection-specific learning rates, correlation coefficients, presynaptic coefficients, postsynaptic coefficients, and momentum coefficients, respectively.

In order to prevent weights from growing too large or too small, we normalize input weights for each output neuron j after each update using the l_2 norm

$$\hat{W}_{i,j}^{t+1} = W_{i,j}^t + \Delta W_{i,j}^t \,, \quad W_{i,j}^{t+1} = \frac{\hat{W}_{i,j}^{t+1}}{\sqrt{\sum_i \hat{W}_{i,j}^{t+1^2}}} \,. \tag{5}$$

The Hebbian layer then returns output $\boldsymbol{W}^{t+1}\boldsymbol{x}^t$, using the updated weights. To regularize the model, we reset the weights for each output neuron $\boldsymbol{W}_j^t$ by drawing randomly from the unit sphere every k examples, where k is a hyperparameter (See **??** for details).

4.3 Experimental Results

4.3.1 Adapting to Static Shift

In Tab 2. we see a clear difference in performance between static shift and no semantic shift, demonstrating the relative difficulty of the two settings.

		No Semantic Shift (Seen Combinations)			Static Shift (Unseen Combinations)		
	Method	Att	Cls	Att+Cls	Att	Cls	Att+Cls
Fixed Model	Resnet(18)	23.34	26.3	10.14	13.63	15.03	1.20
	+IN Pretrain	28.02	<u>34.58</u>	14.57	<u>14.57</u>	<u>20.55</u>	<u>2.31</u>
	+τ-norm	<u>28.05</u>	34.54	<u>14.69</u>	13.09	16.75	1.73
Adaptive Model	ARM-CML	26.07	32.13	12.33	13.89	19.8	1.87
	ARM-BN	27.87	34.23	14.87	13.72	21.93	2.21
	BN	28.31	34.29	14.86	13.79	19.94	2.32
	TTT	28.38	35.71	14.96	<u>14.47</u>	18.19	2.13
	TENT	28.53	34.17	14.80	14.25	19.96	2.28
	Hebb	28.15	36.19	15.32	14.02	**<u>22.30</u>**	**<u>2.38</u>**
	LSTM	<u>28.97</u>	<u>36.35</u>	<u>15.39</u>	14.23	20.63	2.23
	ARM-CML (Cls:25x10)*	28.89	36.34	15.2	14.57	21.04	1.83
	ARM-BN (Cls:25x10)*	29.26	37.01	16.22	14.62	19.71	2.06
	LSTM (Cls:25x10)*	**<u>31.04</u>**	**<u>39.30</u>**	**<u>17.61</u>**	**<u>14.7</u>**	<u>22.21</u>	<u>2.32</u>

Table 2: Att, Cls, and Att+Cls refer to Attribute, Class, and joint Attribute-Class prediction accuracy, respectively. Best performance is bolded and best performance in section (i.e. Fixed, Adaptive, Adaptive (finetuned)) is underlined.
$*$ - MODEL (Cls:25x10) is MODEL finetuned on the training set with a class-shift ordering (25 tasks, 10 occurrences) - See Sec 4.1 and appendix.

ARM approaches underperform all other adaptive methods, even when finetuned on shifting data. Self-Supervised adaptation methods (TTT & TENT) slightly improve performance, but are outperformed by the recurrent and Hebbian method consistently in both settings. Interestingly, fine-tuning the recurrent model on a train set ordering containing continual class shift improves performance when testing without continual shift (i.e. i.i.d. sampling of batches).

4.3.2 ADAPTING TO CONTINUAL CLASS SHIFT

In order to test the effects of catastrophic forgetting and repetition on model performance, we vary the number of occurrences $n \sim [1, 2, 5, 10, 20]$ of each task, while keeping the number of tasks fixed at $k = 25$. Each model is evaluated over 5 different randomly generated test orderings for a given n. We report means in the figures. Model performance was very stable across different orderings, resulting in standard deviations of below 0.1 percentage points (accuracy) in all cases. For continual shift, we additionally evaluate "LSTM-tuned", which is the LSTM method first trained by sampling i.i.d. batches from the train set, then finetuned by simulating continual class shift on the training set (See Sec 4.1, appendix).

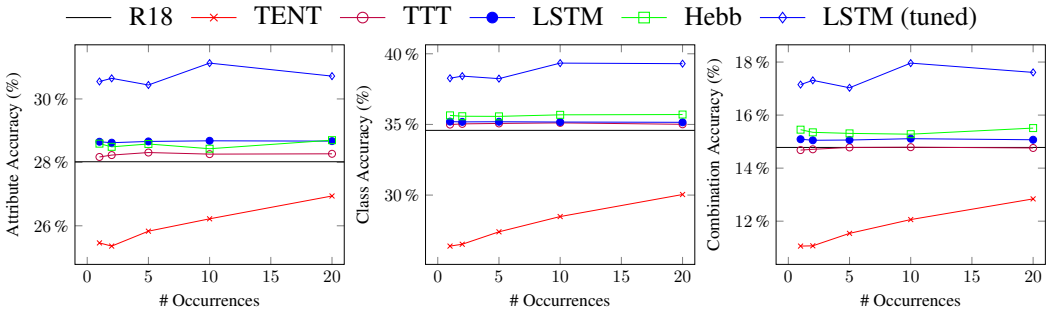

Figure 2: Performance under Class Shift (seen combinations).

TENT significantly underperforms both other adaptive methods as well as the baseline static model. However, its performance improves uniformly as the number of occurrences increases, suggesting that i.i.d. batches are important to the usefulness of the entropy objective for the task. While recurrent and Hebbian methods consistently outperform both the static model and self-supervised adaptation methods on continual shift alone, the improvement over TTT is small, suggesting that rotation prediction is not only a useful task for self-supervised learning (Gidaris et al., 2018), but also for adapting to continually shifting domains. LSTM, Hebb, and TTT, appear to achieve similar performance for all selected values of n occurrences, implying they are not particularly reliant on i.i.d. data for effectiveness. The LSTM method tuned on the training set with continual shift significantly outperforms all other methods for continual shift alone. Future methods for adapting to continual shift may derive significant improvements from simulating shift at test time.

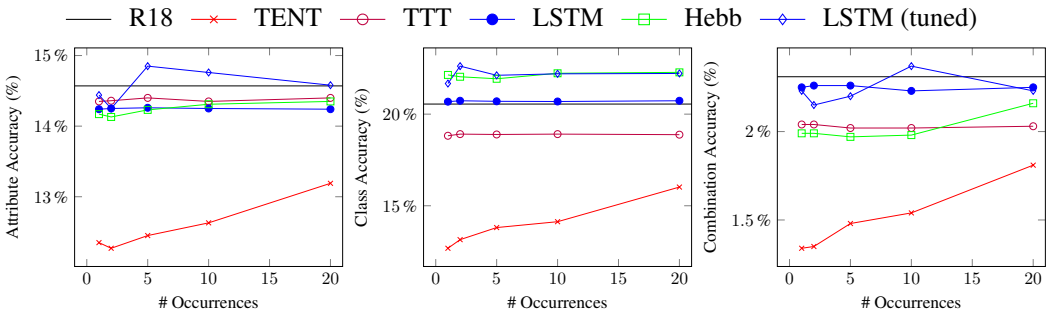

Figure 3: Performance under Class Shift + Static Shift (unseen combinations).

When continual shift is combined with static shift, the results become more complicated. Adaptive methods only improve over the baseline on class prediction (where they see a large gain over TTT), but all models fail to consistently outperform the static method on attribute and combination prediction. All-together, these results suggest that dealing with continual shift and static shift is a much more difficult task than dealing with continual shift alone.

4.3.3 ADAPTING TO CONTINUAL CLASS-ATTRIBUTE SHIFT

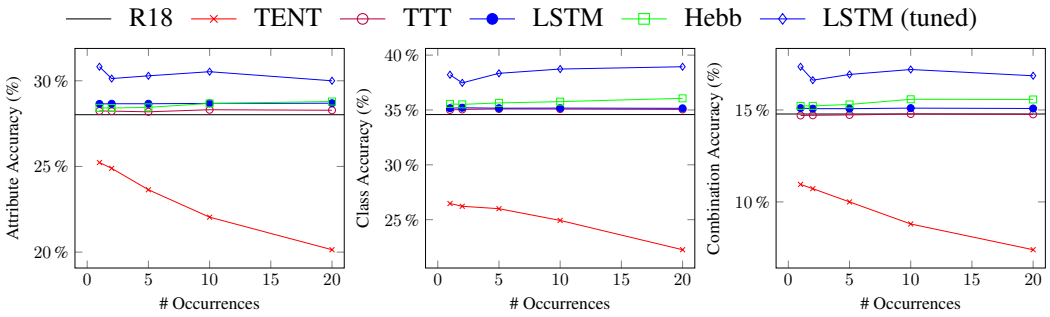

Figure 4: Performance under Class-Attribute Shift (seen combinations).

Without static shift, we see a similar trend for class-attribute shift and for class shift for all methods except for TENT, where we see the opposite trend. TENT uniformly decreases in performance both with and without static shift, suggesting that it relies heavily on co-occurrence of instances with the same class and different attributes for adaptation. When static shift is added, we see a more pronounced version of the effect under class shift, with adaptive methods generally underperforming more with smaller improvements over the static baseline. All-together, results demonstrate that

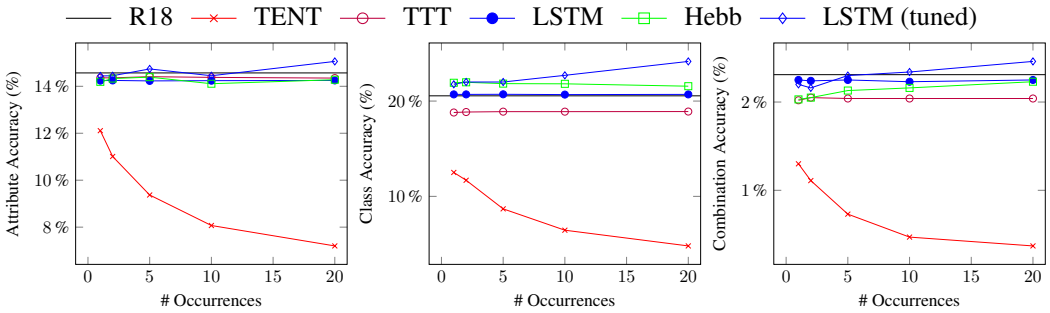

Figure 5: Performance under Class-Attribute Shift + Static Shift (unseen combinations).

class-attribute shift is more difficult than class-shift, both with static shift and without. Additionally it is clear that a recurrent model fine-tuned under continual shift on the train set dominates all other methods when tested on the same parametrization of continual shift on a test set, even if the label combinations were unseen during training. This suggests that learning to adapt to continual shifts at train time can be a strong method for improving performance if they mirror expected test-time shifts.

5 CONCLUSION

In this paper, we introduce the setting of adapting to *Semantic Domain Shift*, aimed at covering many of the unrealistic assumptions in current Domain Adaptation settings. We define *static shift* and *continual shift*. Through extensive experiments, we demonstrate that previous methods for distribution adaptation (both self-supervised and meta-learning based) surprisingly provide little improvement in this setting, sometimes even underperforming a static model that does not change parameters at test time. We introduce models that "learn to adapt" to shifting distributions, outperforming both the static baseline and self-supervised adaptation methods. We demonstrate that recurrence and Hebbian learning can be useful tools for "Learning to adapt". Finally, we provide evidence that training a recurrent model that learns to adapt under continual shift on the training set dominates methods that do not do so if the training shift and test shift are similar.

REFERENCES

Marcin Andrychowicz, Misha Denil, Sergio Gomez Colmenarejo, Matthew W. Hoffman, D. Pfau, Tom Schaul, and N. D. Freitas. Learning to learn by gradient descent by gradient descent. In *NIPS*, 2016.

Pau Panareda Busto and Juergen Gall. Open set domain adaptation. *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 754–763, 2017.

Pau Panareda Busto, Ahsan Iqbal, and Juergen Gall. Open set domain adaptation for image and action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42:413–429, 2020.

Basura Fernando, Amaury Habrard, M. Sebban, and T. Tuytelaars. Unsupervised visual domain adaptation using subspace alignment. *2013 IEEE International Conference on Computer Vision*, pp. 2960–2967, 2013.

Yaroslav Ganin and V. Lempitsky. Unsupervised domain adaptation by backpropagation. *ArXiv*, abs/1409.7495, 2015.

Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations, 2018.

Boqing Gong, Y. Shi, Fei Sha, and K. Grauman. Geodesic flow kernel for unsupervised domain adaptation. *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2066–2073, 2012.

Raghuraman Gopalan, R. Li, and R. Chellappa. Domain adaptation for object recognition: An unsupervised approach. *2011 International Conference on Computer Vision*, pp. 999–1006, 2011.

D. Haviv, Alexander Rivkind, and O. Barak. Understanding and controlling memory in recurrent neural networks. In *ICML*, 2019.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

D. Hebb. The organization of behavior: A neuropsychological theory. 1949.

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9:1735–1780, 1997.

Judy Hoffman, Trevor Darrell, and Kate Saenko. Continuous manifold based adaptation for evolving visual domains. *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 867–874, 2014.

S. Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *ArXiv*, abs/1502.03167, 2015.

Phillip Isola, Joseph J. Lim, and Edward H. Adelson. Discovering states and transformations in image collections. In *CVPR*, 2015.

Vidit Jain and E. Learned-Miller. Online domain adaptation of a pre-trained cascade of classifiers. *CVPR 2011*, pp. 577–584, 2011.

T. Joachims. Transductive inference for text classification using support vector machines. In *ICML*, 1999.

Bingyi Kang, Saining Xie, Marcus Rohrbach, Zhicheng Yan, Albert Gordo, Jiashi Feng, and Yannis Kalantidis. Decoupling representation and classifier for long-tailed recognition. *ArXiv*, abs/1910.09217, 2020.

Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Wei hua Hu, Michihiro Yasunaga, Richard L. Phillips, Sara Beery, Jure Leskovec, Anshul Kundaje, Emma Pierson, Sergey Levine, Chelsea Finn, and Percy Liang. Wilds: A benchmark of in-the-wild distribution shifts. In *ICML*, 2021.

A. Kohn. Visual adaptation: physiology, mechanisms, and functional benefits. *Journal of neurophysiology*, 97 5:3155–64, 2007.

Brian Kulis, Kate Saenko, and Trevor Darrell. What you saw is not what you get: Domain adaptation using asymmetric kernel transforms. *CVPR 2011*, pp. 1785–1792, 2011.

Ananya Kumar, Tengyu Ma, and Percy Liang. Understanding self-training for gradual domain adaptation. In *ICML*, 2020.

Qicheng Lao, Xiang Jiang, Mohammad Havaei, and Yoshua Bengio. Continuous domain adaptation with variational domain-agnostic feature replay. *ArXiv*, abs/2003.04382, 2020.

Yanghao Li, Naiyan Wang, Jianping Shi, Jiaying Liu, and Xiaodi Hou. Revisiting batch normalization for practical domain adaptation, 2016.

H. Liu, Mingsheng Long, J. Wang, and Yu Wang. Learning to adapt to evolving domains. In *NeurIPS*, 2020a.

Ziwei Liu, Zhongqi Miao, Xingang Pan, Xiaohang Zhan, Dahua Lin, Stella X. Yu, and Boqing Gong. Open compound domain adaptation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020b.

Thomas Miconi. Learning to learn with backpropagation of hebbian plasticity, 2016.

Thomas Miconi, Aditya Rawal, J. Clune, and K. Stanley. Backpropamine: training self-modifying neural networks with differentiable neuromodulated plasticity. *ArXiv*, abs/2002.10585, 2019.

C. Morgan. Animal intelligence. *Nature*, 26:523–524.

Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010.

E. Najarro and S. Risi. Meta-learning through hebbian plasticity in random networks. *ArXiv*, abs/2007.02686, 2020.

Cuong V Nguyen, A. Achille, Michael Lam, Tal Hassner, V. Mahadevan, and Stefano Soatto. Toward understanding catastrophic forgetting in continual learning. *ArXiv*, abs/1908.01091, 2019.

Y. Noguchi, K. Inui, and R. Kakigi. Temporal dynamics of neural adaptation effect in the human visual ventral stream. *The Journal of Neuroscience*, 24:6283 – 6290, 2004.

G. I. Parisi, Ronald Kemker, Jose L. Part, Christopher Kanan, and S. Wermter. Continual lifelong learning with neural networks: A review. *Neural networks : the official journal of the International Neural Network Society*, 113:54–71, 2019.

Joaquin Quionero-Candela, Masashi Sugiyama, Anton Schwaighofer, and Neil Lawrence. Dataset shift in machine learning. 2009.

Olga Russakovsky, J. Deng, Hao Su, J. Krause, S. Satheesh, S. Ma, Zhiheng Huang, A. Karpathy, A. Khosla, Michael S. Bernstein, A. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115:211–252, 2015.

Kate Saenko, Brian Kulis, Mario Fritz, and Trevor Darrell. Adapting visual category models to new domains. In *ECCV*, 2010.

Shibani Santurkar, D. Tsipras, and A. Madry. Breeds: Benchmarks for subpopulation shift. *arXiv: Computer Vision and Pattern Recognition*, 2021.

Shira Sardi, R. Vardi, Yuval Meir, Yael Tugendhaft, Shiri Hodassman, A. Goldental, and I. Kanter. Brain experiments imply adaptation mechanisms which outperform common ai learning algorithms. *Scientific Reports*, 10, 2020.

J. Schmidhuber. Learning to control fast-weight memories: An alternative to dynamic recurrent networks. *Neural Computation*, 4:131–139, 1992.

A. Soltoggio, Peter Dürr, C. Mattiussi, and D. Floreano. Evolving neuromodulatory topologies for reinforcement learning-like problems. *2007 IEEE Congress on Evolutionary Computation*, pp. 2471–2478, 2007.

Y. Sun, X. Wang, Zhuang Liu, J. Miller, Alexei A. Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *ICML*, 2020.

Vithursan Thangarasa, Thomas Miconi, and Graham W. Taylor. Enabling continual learning with differentiable hebbian plasticity, 2020.

A. Torralba and Alexei A. Efros. Unbiased look at dataset bias. *CVPR 2011*, pp. 1521–1528, 2011.

Gido M. van de Ven and A. Tolias. Three scenarios for continual learning. *ArXiv*, abs/1904.07734, 2019.

V. Vapnik. Transductive inference and semi-supervised learning. In *Semi-Supervised Learning*, 2006.

Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization, 2021.

Hao Wang, Hao He, and D. Katabi. Continuously indexed domain adaptation. In *ICML*, 2020.

Zuxuan Wu, Xin Wang, Joseph E. Gonzalez, T. Goldstein, and L. Davis. Ace: Adapting to changing environments for semantic segmentation. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 2121–2130, 2019.

Markus Wulfmeier, A. Bewley, and I. Posner. Incremental adversarial domain adaptation for continually changing environments. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1–9, 2018.

Marvin Zhang, Henrik Marklund, Abhishek Gupta, Sergey Levine, and Chelsea Finn. Adaptive risk minimization: A meta-learning approach for tackling group shift. *ArXiv*, abs/2007.02931, 2020.

S. Zhao, Xiangyu Yue, Shanghang Zhang, Bo Li, Han Zhao, B. Wu, Ravi Krishna, J. Gonzalez, A. Sangiovanni-Vincentelli, S. Seshia, and K. Keutzer. A review of single-source deep unsupervised visual domain adaptation. *IEEE transactions on neural networks and learning systems*, PP, 2020.