Test-Time Adaptation with State-Space Models

Mona Schirmer [1] **Dan Zhang** [2] **Eric Nalisnick** [3]

Abstract

Distribution shifts between training and test data are all but inevitable over the lifecycle of a deployed model and lead to performance decay. Adapting the model can hopefully mitigate this drop in performance. Yet, adaptation is challenging since it must be unsupervised: we usually do not have access to any labeled data at test time. In this paper, we propose a probabilistic state-space model that can adapt a deployed model subjected to distribution drift. Our model learns the dynamics induced by distribution shifts on the last set of hidden features. Without requiring labels, we infer time-evolving class prototypes that serve as a dynamic classification head. Moreover, our approach is lightweight, modifying only the model's last linear layer. In experiments on real-world distribution shifts and synthetic corruptions, we demonstrate that our approach performs competitively with methods that require back-propagation and access to the model backbone. Our model especially excels in the case of small test batches—the most difficult setting.

1. Introduction

Predictive models often have an 'expiration date.' Real-world applications tend to exhibit distribution drift, meaning that the data points seen at test time are drawn from a distribution that is different than the training data's. Moreover, the test distribution usually becomes more unlike the training distribution as time goes on. An example of this is with recommendation systems: trends change, new products are released, old products are discontinued, etc. Unless a model is updated, its ability to make accurate predictions will expire, requiring the model to be taken offline and re-trained. Every iteration of this model life-cycle can be expensive and time consuming. Allowing models to remain 'fresh' for as long as possible is thus an open and consequential problem.

In this work, we propose *State-space Test-time ADaptation* (STAD), a method that delays the failure of a deployed model by performing unsupervised adaptation at test time. We perform this updating by modeling the dynamics of the parameters at a neural network's final layer, making our approach widely applicable and computationally lightweight. Specifically, we use a state-space model (SSM) to track how the weight vectors in the final layer—where each vector corresponds to a class—evolve under distribution drift. To generate predictions for the newly-acquired batch of test points, we use the SSM's fitted cluster means as the model's updated parameters. We focus on natural data shifts caused by a gradually changing environment rather than noise-based shift, which has been the focus of previous work (Wang et al., 2021; 2022). Our contributions are as follows,

- In Section 3.2, we present STAD, a state-space model to learn the dynamics of how a classifier's last-layer weights evolve under distribution shift, without access to any labels. No previous work has explicitly modeled these dynamics, which we demonstrate is crucial via an ablation study.
- In Sections 3.2 and 3.3, we provide two implementations of STAD—one using Gaussian distributions and one using von Mises-Fisher distributions. Each represents a different assumption about the geometry of the model's last-layer representations.
- In Section 5, we apply STAD to real-world (or 'natural') shifts, showing improvements over state-of-the-art adaptation methods such as *TENT* (Wang et al., 2021) and *CoTTA* (Wang et al., 2022). Previous work has mostly focused on synthetic noise-based shifts, and our method is competitive on this task as well.

2. Problem Setting

Data & Model We focus on the traditional setting of multi-label classification, where $\mathcal{X} \subseteq \mathbb{R}^D$ denotes the input (feature) space and $\mathcal{Y} \subseteq \{1, \dots, K\}$ denotes the label space. Let $\mathbf{x}$ and y be random variables and $\mathbb{P}(\mathbf{x}, \mathrm{y}) = \mathbb{P}(\mathbf{x})\,\mathbb{P}(\mathrm{y}|\mathbf{x})$ the unknown source data distribution. We assume $\boldsymbol{x} \in$

[1]UvA-Bosch Delta Lab, University of Amsterdam [2]Bosch Center for AI & University of Tübingen [3]Johns Hopkins University. Correspondence to: Mona Schirmer <m.c.schirmer@uva.nl>.

Accepted by the Structured Probabilistic Inference & Generative Modeling workshop of ICML 2024, Vienna, Austria.

$\mathcal{X}$ and $y \in \mathcal{Y}$ are realisations of $\mathbf{x}$ and y. The goal of classification is to find a mapping f_θ, with parameters θ, from the input space to the label space $f_\theta : \mathcal{X} \to \mathcal{Y}$. Fitting the classifier f_θ is usually accomplished by minimizing an appropriate loss function (e.g. log loss). Yet, our method is agnostic to how f_θ is trained and therefore easy to use with, for instance, a pre-trained model that has been downloaded from the web.

Distribution Drift and Unsupervised Adaptation In predictive modeling, a classifier will almost always become 'stale,' as the test conditions inevitably change from those observed during training. In fact, the model's presence in the world can cause this change: a model that predicts when someone is likely to be infected with a contagious disease will, if effective and widely adopted, reduce the prevalence of that disease. Thus, we want our models to be 'fresh' for as long as possible and continue to make accurate predictions, in spite of data drift. More formally, let the data at test-time t be sampled from a distribution $\mathbb{Q}_t(\mathbf{x}, \mathrm{y}) = \mathbb{Q}_t(\mathbf{x})\, \mathbb{Q}_t(\mathrm{y}|\mathbf{x})$ such that $\mathbb{Q}_t(\mathbf{x}, \mathrm{y}) \neq \mathbb{P}(\mathbf{x}, \mathrm{y})\ \forall t > 0$. Of course, we do not observe labels at test time, and hence we observe only a batch of features $\mathbf{X}_t = \{\mathbf{x}_{1,t}, \ldots, \mathbf{x}_{N,t}\}$, where $\mathbf{x}_{n,t} \sim \mathbb{Q}_t(\mathbf{x})$ (i.i.d.). Given the t-th batch of features $\mathbf{X}_t$, the goal is to adapt f_θ, forming a new set of parameters θ_t such that f_{θ_t} has better predictive performance on $\mathbf{X}_t$ than f_θ would have. Since we can only observe features, we assume that the distribution shift must at least take the form of *covariate shift*: $\mathbb{Q}_t(\mathbf{x}) \neq \mathbb{P}(\mathbf{x})\ \forall t > 0$. $\mathbb{Q}_t(\mathrm{y}|\mathbf{x})$ could shift as well, but this will be impossible to detect in our assumed setting. If $\mathbb{Q}_t(\mathrm{y}|\mathbf{x})$ does shift, it must do so modestly, as the classifier will become completely invalid if the relationship between features and labels changes to a severe degree. This setting is known as *continual test-time adaptation* (CTTA) (Wang et al., 2022; Gong et al., 2022; Press et al., 2024a; Niu et al., 2022; Song et al., 2023; Döbler et al., 2023).

3. Modelling the Dynamics of Distribution Shifts in Representation Space

We now present our method: the core idea is that CTTA can be done by tracking how distribution shift affect the model's representations. We employ linear state-space models (SSMs) to capture how test points evolve and drift. The SSM's cluster representations then serve as an adaptive classification head that evolves with the non-stationarity of the distribution drift. In Section 3.2, we first introduce the general model and then in Section 3.3, we propose an implementation that leverages the von-Mises-Fisher distribution to model hyperspherical features.

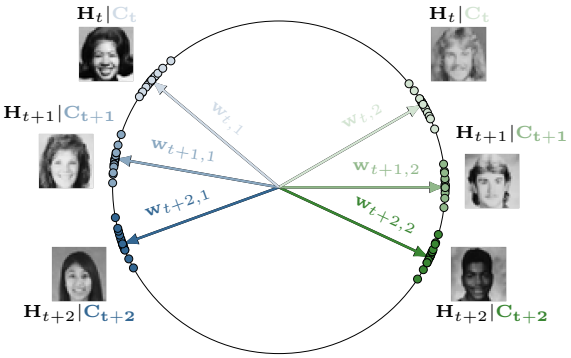

Figure 1. Illustration for the 2D Unit Sphere: Fashion trends and demographic changes lead to a shift in the representations $\mathbf{H}$. STAD adapts to the distribution shift by inferring dynamic class prototypes $\mathbf{w}_{t+2,1}$ and $\mathbf{w}_{t+2,2}$ for test time point $t+2$ observing only neural representations $\mathbf{H}_{t+2}$ and no labels.

3.1. Modeling Shift in Representation Space

Whereas previous work has mostly attempted to adapt all levels of a neural network, we take the alternative approach of adapting only the last layer. Let the classifier f_θ be a neural network having L total layers. We will treat the first $L-1$ as a black box, assuming they transform the original feature vector $\mathbf{x}$ into a new (lower dimensional) representation, which we denote as $\mathbf{h}$. The original classifier then maps these representations to the classes as: $\mathbb{E}[y|\mathbf{h}] = \texttt{softmax}_y(\mathbf{W}_0\mathbf{h})$, where $\texttt{softmax}_y(\cdot)$ denotes the dimension of the softmax's output corresponding to the y-th label index and $\mathbf{W}_0$ are the last-layer weights. As $\mathbf{W}_0$ will only be valid for representations that are similar to the training data, we will discard these parameters when performing CTTA, learning new parameters $\mathbf{W}_t$ for the t-th time step. These new parameters will be used to generate the adapted predictions through the same link function: $\mathbb{E}[y|\mathbf{h}] = \texttt{softmax}_y(\mathbf{W}_t\mathbf{h})$.

In the setting of CTTA, we observe a batch of features $\mathbf{X}_t$. Passing them through the model yields corresponding representations $\mathbf{H}_t$, and this will be the 'data' used for the probabilistic model we will describe below. Specifically, we will model how the representations change from $\mathbf{H}_t$ to $\mathbf{H}_{t+1}$. Operating on this last set of hidden representations has several benefits. Firstly, test-time adaptation is fast as $\mathbf{H}_t$ is relatively low dimensional and no backpropagation is necessary. Secondly, $\mathbf{H}_t$ could be provided by a foundation model accessed through a black-box API. Lastly, recall that we do not have the labels necessary to fit a complex classifier on $\mathbf{H}_t$ directly. In turn, for $\mathbf{H}_t$ to be useful, these representations must exhibit a clear structure (e.g. clustering) that reflects the classes. This assumption can lead to diagnostics that determine if we should perform CTTA or not, as we will see in the experiments.

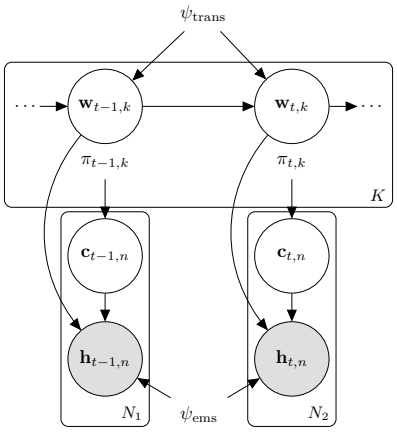

Figure 2. Graphical model: Representations are modeled with a dynamic mixture model. Latent class prototypes $\mathbf{w}_{t,k}$ evolve at each time step, cluster assignments $\mathbf{c}_{t,n}$ determine class membership of each neural representations $\mathbf{h}_{t,n}$.

3.2. A Probabilistic Model of Shift Dynamics

We now describe our general method for adaptive last-layer parameters. We assume that, while the representations $\mathbf{H}$ are changing, they are still maintaining some class structure in the form of clusters. Our model will seek to track this structure as it evolves over time. Using latent variables $\mathbf{w}_{t,k}$, we will assume each representation is drawn conditioned on K latent vectors: $\boldsymbol{h}_{t,n} \sim p\left(\mathbf{h}_t|\mathbf{w}_{t,1}, \ldots, \mathbf{w}_{t,K}\right)$, where K is equal to the number of classes in the prediction task. After fitting the unsupervised model, the K latent vectors will be stacked to create $\mathbf{W}_t$, the last-layer weights of the adapted predictive model (as introduced in 3.1). For the intuition of the approach, see Figure 1. The blue and red clusters represent each class of a binary problem. As the distribution shifts from time step t to $t+1$, both classes migrate downward, towards the equator of the circle. Our SSM aims to track this movement so that classification performance can still be preserved. Why the representations are restricted to the sphere will become clear in Section 3.3. We now move on to a technical description.

Notation and Variables Let $\mathbf{H}_t = (\mathbf{h}_{t,1}, \ldots, \mathbf{h}_{t,N_t}) \in \mathbb{R}^{D \times N_t}$ denote the neural representations for N_t data points at test time t. Let $\mathbf{W}_t = (\mathbf{w}_{t,1}, \ldots, \mathbf{w}_{t,K}) \in \mathbb{R}^{D \times K}$ denote the K weight vectors at test time t. As discussed above, the weight vector $\mathbf{w}_{t,k}$ can be thought of as a latent prototype for class k at time t. We denote with $\mathbf{C}_t = (\mathbf{c}_{t,1}, \ldots, \mathbf{c}_{t,N_t}) \in \{0,1\}^{K \times N_t}$ the N_t one-hot encoded latent class assignment vectors $\mathbf{c}_{t,n} \in \{0,1\}^K$ at time t. The k-th position of $\mathbf{c}_{t,n}$ is denoted with $\mathrm{c}_{t,n,k}$ and is 1 if $\mathbf{h}_{t,n}$ belongs to class k and 0 otherwise. Like in standard (static) mixture models (Bishop and Nasrabadi, 2006), the prior of the latent class assignments $p(\mathbf{c}_{t,n})$ is a categorical distribution, $p(\mathbf{c}_{t,n}) = \texttt{Cat}(\boldsymbol{\pi}_t)$ with $\boldsymbol{\pi}_t = (\pi_{t,1}, \ldots, \pi_{t,K}) \in [0,1]^K$ and $\sum_{k=1}^K \pi_{t,k} = 1$. The mixing coefficient $\pi_{t,k}$ gives the a priori probability of class k at time t and can be interpreted as the class proportions.

Dynamics Model We model the evolution of the K prototypes $\mathbf{W}_t = (\mathbf{w}_{t,1}, \ldots, \mathbf{w}_{t,K})$ with K independent Markov processes. The resulting transition model is then:

$$p(\mathbf{W}_t|\mathbf{W}_{t-1}, \psi^{\text{trans}}) = \prod_{k=1}^{K} p(\mathbf{w}_{t,k}|\mathbf{w}_{t-1,k}, \psi^{\text{trans}}), \quad (1)$$

where ψ^{trans} denote the parameters of the transition density, notably the transition noise. At each time step, the feature vectors $\mathbf{H}_t$ are generated by a mixture distribution over the K classes,

$$p(\mathbf{H}_t|\mathbf{W}_t, \psi^{\text{ems}}) = \prod_{n=1}^{N_t} \sum_{k=1}^{K} \pi_{t,k} \cdot p(\mathbf{h}_{t,n}|\mathbf{w}_{t,k}, \psi^{\text{ems}}). \quad (2)$$

where ψ^{ems} are the emission parameters. We thus assume at each time step a standard mixture model over the K classes where the class prototype $\mathbf{w}_{t,k}$ defines the latent class center and $\pi_{t,k}$ the mixture weight for class k. The joint distribution of representations, prototypes and class assignments can be factorised as follows,

$$\begin{aligned}
&p(\mathbf{H}_{1:T}, \mathbf{W}_{1:T}, \mathbf{C}_{1:T}) \\
&= p(\mathbf{W}_1) \prod_{t=1}^{T} p(\mathbf{C}_t) p(\mathbf{H}_t|\mathbf{W}_t, \mathbf{C}_t, \psi^{\text{ems}}) \\
&\quad \prod_{t=2}^{T} p(\mathbf{W}_t|\mathbf{W}_{t-1}, \psi^{\text{trans}}) \\
&= \prod_{k=1}^{K} p(\mathbf{w}_{1,k}) \prod_{t=1}^{T} \prod_{n=1}^{N_t} p(\mathbf{c}_{t,n}) \prod_{k=1}^{K} p(\mathbf{h}_{t,n}|\mathbf{w}_{t,k}, \psi^{\text{ems}})^{\mathrm{c}_{t,n,k}} \\
&\quad \prod_{t=2}^{T} \prod_{k=1}^{K} p(\mathbf{w}_{t,k}|\mathbf{w}_{t-1,k}, \psi^{\text{trans}}).
\end{aligned} \quad (3)$$

We use the notation $\mathbf{H}_{1:T} = \{\mathbf{H}_t\}_{t=1}^T$ to denote the representation vectors $\mathbf{H}_t$ for all time steps T and analogously for $\mathbf{W}_{1:T}$ and $\mathbf{C}_{1:T}$. The model's plate diagram is depicted in Figure 2. The representation $\mathbf{H}_{1:T}$ are the observed variables and are determined by the latent class prototypes $\mathbf{W}_{1:T}$ and the latent class assignments $\mathbf{C}_{1:T}$.

Posterior Inference & Adapted Predictions The primary goal is to update the class prototypes $\mathbf{W}_t$ with the information obtained by the N_t representations of test time t. At each test time t, we are thus interested in the posterior distribution of the prototypes $p(\mathbf{W}_t|\mathbf{H}_{1:t})$. Once $p(\mathbf{W}_t|\mathbf{H}_{1:t})$ is known, we can update the classification

weights with the new posterior mean. The class weights $\mathbf{W}_t$ and class assignments $\mathbf{C}_t$ can be inferred using the Expectation-Maximization (EM) algorithm. In the E-step, we compute $p(\mathbf{W}_{1:T}\mathbf{C}_{1:T}|\mathbf{H}_{1:T})$. In the M-Step, we maximize the expected complete-data log likelihood with respect to the model parameters:

$$\phi^* = \arg\max_{\phi} \mathbb{E}_{p(\mathbf{W},\mathbf{C}|\mathbf{H})}\big[\log p(\mathbf{H}_{1:T}, \mathbf{W}_{1:T}, \mathbf{C}_{1:T})\big], \tag{4}$$

where ϕ denotes the parameters of the transition and emission density as well as the mixing coefficients, $\phi = \{\psi^{\text{trans}}, \psi^{\text{ems}}, \boldsymbol{\pi}_{1:T}\}$. After one optimization step, we collect the K class prototypes into a matrix $\mathbf{W}_t$. Using the same hidden representations used to fit $\mathbf{W}_t$, we generate the predictions using the original predictive model's softmax parameterization:

$$y_{t,n} \sim \texttt{Cat}\left(\mathrm{y}_{t,n};\ \texttt{softmax}(\mathbf{W}_t\mathbf{h}_{t,n})\right) \tag{5}$$

where $y_{t,n}$ denotes a prediction sampled for the representation vector $\mathbf{h}_{t,n}$. Note that adaptation can be performed in an online fashion by optimizing Equation (4) incrementally considering points up to point t. To omit computing the complete-data log likelihood for an increasing sequence as time goes on, we employ a sliding window approach.

Gaussian Model The simplest parametric form for the transition and emissions models is Gaussian. The resulting model can be seen as a mixture of K Kalman filters (KFs). For posterior inference, thanks to the linearity and Gaussian assumptions, the posterior expectation $\mathbb{E}_{p(\mathbf{W},\mathbf{C}|\mathbf{H})}[\cdot]$ in Equation (4) can be computed analytically using the well known KF predict, update and smoothing equations (Calabrese and Paninski, 2011; Bishop and Nasrabadi, 2006). However, the closed-form computations come at a cost as they involve matrix inversions of dimensionality $D \times D$. Moreover, the parameter size scales $K \times D^2$, risking overfitting and consuming substantial memory. These are limitations of the Gaussian formulation making it costly for high-dimensional feature spaces and impractical in low resource environments requiring instant predictions. In the next section, we discuss a model for spherical features that elegantly circumvents the limitations of a fully parameterized Gaussian formulation.

3.3. Von Mises-Fisher Model for Hyperspherical Features

Choosing Gaussian densities for the transition and emission models, as discussed above, assumes the representation space follows an Euclidean geometry. However, prior work has shown that assuming the hidden representations lie on the unit *hypersphere* results in a better inductive bias for OOD generalization (Mettes et al., 2019; Bai et al., 2024). This is due to the norms of the representations being biased by in-domain information such as class balance, making angular distances a more reliable signal of class membership in the presence of distribution shift (Mettes et al., 2019; Bai et al., 2024). We too employ the hyperspherical assumption by normalizing the hidden representations such that $||\mathbf{h}||_2 = 1$ and modeling them with the *von Mises-Fisher* (vMF) distribution (Mardia and Jupp, 2009),

$$\texttt{vMF}(\mathbf{h}; \boldsymbol{\mu}_k, \kappa) = C_D(\kappa) \exp\left\{\kappa \cdot \boldsymbol{\mu}_k^T \mathbf{h}\right\} \tag{6}$$

where $\boldsymbol{\mu}_k \in \mathbb{R}^D$ with $||\boldsymbol{\mu}_k||_2 = 1$ denotes the mean direction of class k, $\kappa \in \mathbb{R}^+$ the concentration parameter, and $C_D(\kappa)$ the normalization constant. High values of κ imply larger concentration around $\boldsymbol{\mu}_k$. The vMF distribution is proportional to a Gaussian distribution with isotropic variance and unit norm. While previous work (Mettes et al., 2019; Ming et al., 2023; Bai et al., 2024) has mainly explored training objectives to encourage latent representations to be vMF-distributed, we apply Equation (6) to model the evolving representations.

Hyperspherical State-Space Model Returning to the SSM given above, we specify both transition and emission models as vMF distributions,

$$p(\mathbf{W}_t|\mathbf{W}_{t-1}) = \prod_{k=1}^{K} \texttt{vMF}(\mathbf{w}_{t,k}|\mathbf{w}_{t-1,k}, \kappa^{\text{trans}}) \tag{7}$$

$$p(\mathbf{H}_t|\mathbf{W}_t) = \prod_{n=1}^{N_t} \sum_{k=1}^{K} \pi_{t,k} \texttt{vMF}(\mathbf{h}_{t,n}|\mathbf{w}_{t,k}, \kappa^{\text{ems}}) \tag{8}$$

The parameter size of the vMF formulation only scales linearly with the feature dimension, i.e. $\mathcal{O}(DK)$ instead of $\mathcal{O}(D^2K)$ as for the Gaussian case. The noise parameters, $\kappa^{\text{trans}}, \kappa^{\text{ems}}$ are scalar values, and we shall use the reduced parameter size to experiment with class conditioned noise parameters $\kappa_k^{\text{trans}}, \kappa_k^{\text{ems}}, k = 1, \ldots, K$ in Section 5.

Posterior Inference Unlike in the linear Gaussian case, the vMF distribution is not closed under marginalization. Consequentially, the posterior distribution required for the expectation in Equation (4), $p(\mathbf{W}_{1:T}\mathbf{C}_{1:T}|\mathbf{H}_{1:T})$, cannot be obtained in closed form. We employ a variational EM objective, approximating the posterior with mean-field variational inference, following Gopal and Yang (2014):

$$q(\mathbf{w}_{t,k}) = \texttt{vMF}(\,\cdot\,; \boldsymbol{\rho}_{t,k}, \gamma_{t,k}) \quad q(\mathbf{c}_{n,t}) = \texttt{Cat}(\,\cdot\,; \boldsymbol{\lambda}_{n,t}) \tag{9}$$

The variational distribution $q(\mathbf{W}, \mathbf{C})$ factorizes over n, t, k and the objective from Equation (4) becomes $\arg\max_\phi \mathbb{E}_{q(\mathbf{W},\mathbf{C})}\big[\log p(\mathbf{H}_{1:T}, \mathbf{W}_{1:T}, \mathbf{C}_{1:T})\big]$. More details as well as the full maximisation steps for $\phi = \{\kappa^{\text{trans}}, \kappa^{\text{ems}}, \{\pi_{t,k}\}_{t=1}^T\}_{k=1}^K\}$ can be found in Appendix B. Notably, posterior inference for this model is much more scalable than the Gaussian case, having operations that are linear in the dimensionality rather than cubic.

Recovering the Softmax Predictive Distribution In addition to the inductive bias that is beneficial under distribution shift, using the vMF distribution has an additional desirable property: classification via the cluster assignments is equivalent to the original softmax-parameterized classifier. The equivalence is exact under the assumption of equal class proportions and sharing κ across classes:

$$\begin{aligned} & p(\mathrm{c}_{t,n,k} = 1 | \mathbf{h}_{t,n}, \mathbf{w}_{t,1}, \ldots, \mathbf{w}_{t,K}, \kappa^{\text{ems}}) \\ &= \frac{\texttt{vMF}(\mathbf{h}_{t,n}; \mathbf{w}_{t,k}, \kappa^{\text{ems}})}{\sum_{j=1}^{K} \texttt{vMF}(\mathbf{h}_{t,n}; \mathbf{w}_{t,j}, \kappa^{\text{ems}})} \\ &= \frac{C_D(\kappa^{\text{ems}}) \exp\left\{\kappa^{\text{ems}} \cdot \mathbf{w}_{t,k}^T \mathbf{h}_{t,n}\right\}}{\sum_{j=1}^{K} C_D(\kappa^{\text{ems}}) \exp\left\{\kappa^{\text{ems}} \cdot \mathbf{w}_{t,j}^T \mathbf{h}_{t,n}\right\}} \\ &= \texttt{softmax}\left(\kappa^{\text{ems}} \cdot \mathbf{W}_t^T \mathbf{h}_{t,n}\right), \end{aligned} \tag{10}$$

which is equivalent to a softmax with temperature-scaled logits, with the temperature set to $1/\kappa^{\text{ems}}$. Temperature scaling only affects the probabilities, not the modal class prediction. If using class-specific κ^{ems} values and assuming imbalanced classes, then these terms show up as class-specific bias terms: $p(\mathrm{c}_{t,n,k} = 1 | \mathbf{h}_{t,n}, \mathbf{w}_{t,1}, \ldots, \mathbf{w}_{t,K}, \kappa_1^{\text{ems}}, \ldots, \kappa_K^{\text{ems}}) \propto \exp\left\{\kappa_k^{\text{ems}} \cdot \mathbf{w}_{t,k}^T \mathbf{h}_{t,n} + \log C_D(\kappa_k^{\text{ems}}) + \log \pi_{t,k}\right\}$ where $C_D(\kappa_k^{\text{ems}})$ is the vMF's normalization constant and $\pi_{t,k}$ is the mixing weight.

4. Related Work

Filtering for Deep Learning Traditional filtering models, and the Kalman filter (Kalman, 1960) in particular, have recently found use in deep learning as a principled way of updating a latent state with new information. In sequence modelling, filter-based architectures are used to learn the latent state of an observation trajectory in both discrete (Krishnan et al., 2015; Karl et al., 2017; Fraccaro et al., 2017; Becker et al., 2019) and continuous time (Schirmer et al., 2022; Ansari et al., 2023; Zhu et al., 2023). However, here, the filtering model mimics the dynamics of individual observation sequences while we are interested in modeling the dynamics of the data stream as a whole. Chang et al. (2023) and Titsias et al. (2023) employ Kalman filters in a supervised online learning setting to update neural network weights to a non-stationary data stream. Like our method, Titsias et al. (2023) infers the evolution of the linear classification head. However, (Chang et al., 2023; Titsias et al., 2023) rely on labels to update the latent state of the weights after prediction. In contrast, our weight adaptation is entirely label-free.

Test-Time Adaptation Maintaining reliable predictions under distribution shift at test time has driven several research directions such as continual learning (De Lange et al., 2021) and domain adaptation (Patel et al., 2015; Wilson and Cook, 2020). Our setting falls into test-time adaptation, where the goal is to adapt the source-trained model given only access to the unlabeled target data (Liang et al., 2023; Yu et al., 2023). A simple yet effective approach is to keep updating the batch normalization (BN) statistics at test time (Schneider et al., 2020; Nado et al., 2020). Based on this insight, a common strategy is to learn the parameters of the BN layer during test time (Schneider et al., 2020; Nado et al., 2020; Wang et al., 2021; Gong et al., 2022; Niu et al., 2022; Press et al., 2024a). For instance, TENT (Wang et al., 2021) learns the BN parameters by minimizing entropy on the predicted target labels. A variation of the setting arises when the test distribution itself changes over time, a more realistic scenario studied by continual test-time adaptation. The main challenge in this paradigm is to ensure adaptability while preventing catastrophic forgetting of the source distribution. To mitigate this trade off, strategies involve episodic resetting to the source parameters (Wang et al., 2022; Press et al., 2024a) or test sample selection (Niu et al., 2022). Nonetheless, it has been observed that many strategies still experience performance degradation after extended periods of adaptation (Niu et al., 2022; Gong et al., 2022; Wang et al., 2022; Press et al., 2024a).

5. Experiments

Our experimental goal is to demonstrate that STAD effectively models *natural temporal drifts* in the test-time covariates. This distinguishes our work from previous CTTA methods, which focus on domain shifts or shifts induced by corruption noise but do not consider natural evolution of time. Precisely, our setting of interest comprises (1) real-world shifts that (2) occur gradually over time and (3) lead to performance decay. Though ubiquitous in practice, systematic evaluation procedures for natural temporal drifts have only been considered fairly recently (Yao et al., 2022). We use the evaluation protocol of the Wild-Time benchmark suite (Yao et al., 2022) to assess the adaptation performance of our model in this setting of natural shifts. In Section 5.1, we demonstrate that our method excels on natural temporal drifts yielding significant performance improvements in settings in which existing CTTA methods collapse. We show the importance of explicitly modeling shift dynamics via an ablation study in Section 5.2. Finally, in Section 5.3, we investigate limitations of our model and show how to diagnose failures ahead of time.

Datasets We consider two image classification datasets exposed to natural temporal drifts. In addition to our main setting of interest, we also present results on a classic corruption dataset.

- **Yearbook** (Ginosar et al., 2015): a dataset of portraits

of American high school students taken across eight decades. Data shift in the students' visual appearance is introduced by changing beauty standards, group norms, fashion trends, and demographic changes. We use the Wild-Time (Yao et al., 2022) pre-processing and evaluation procedure resulting into 33,431 images from 1930 to 2013. Each 32×32 pixel, grey-scaled image is associated with the student's gender as a binary target label. Images from 1930 to 1969 are used for training; the remaining years from 1970 to 2013 for testing.

- **FMoW-Time**: the *functional map of the world* (FMoW) dataset (Koh et al., 2021) maps 224×224 RGB satellite images to one of 62 land use categories. Distribution shift is introduced by technical advancement and economic growth changing how humans make use of land over time. FMoW-Time (Yao et al., 2022) is an adaptation from FMoW-WILDS (Koh et al., 2021; Christie et al., 2018) that splits a total of 141,696 images into a training time period (2002 - 2012) and a testing time period (2013 - 2017).
- **CIFAR-10-C**: a dataset derived from CIFAR-10, to which 15 corruption / noise types are applied with 5 severity levels to introduce gradual distribution shift (Hendrycks and Dietterich, 2019). We increase the corruption severity starting from the lowest level (severity 1) to the most sever corruption (severity 5). This results in a test stream of $5 \times 10,000$ images for each corruption type. Since our goal is mimicking gradual distribution shifts, we are not interested in switching between images of different corruption types, as previous work has done (Wang et al., 2022; 2021).

Source Architectures We use a variety of source architectures to demonstrate the model-agnostic nature of our method. They vary in parameter counts, backbone architecture and dimensionality of the representation space.

- **CNN**: We employ the four-block convolutional neural network trained by (Yao et al., 2022) to perform the binary gender prediction on the yearbook dataset. Presented results are averages over three different random training seeds. The dimension of the latent representation space is 32.
- **DenseNet**: For FMoW-Time, we follow the backbone choice of (Koh et al., 2021; Yao et al., 2022) and use DenseNet121 (Huang et al., 2017) for the land use classification task. Weights for three random trainings seeds are provided by (Yao et al., 2022). The latent representation dimension is 1024.
- **WideResNet**: For the CIFAR-10 experiment, we follow (Song et al., 2023; Wang et al., 2021) and use the pre-trained WideResNet-28 (Zagoruyko and Komodakis, 2016) model from RobustBench (Croce et al., 2021). The latent representation have 512 dimensions.

Baselines Despite the source model, we compare against three baselines suitable for an unsupervised, continuously-changing test stream. These baselines cover the most dominant paradigms in CTTA: collecting and adapting normalization statistics, entropy minimization and anti-collapse mechanics.

- **Source Model**: the un-adapted original model.
- **BatchNorm (BN) Adaptation** (Schneider et al., 2020; Nado et al., 2020): aims to adapt the source model to distributions shift by collecting normalization statistics (mean and variance) of the test data.
- **Test Entropy Minimization (TENT)** (Wang et al., 2021): goes one step further and optimizes the BN transformation parameters (scale and shift) by minimizing entropy on test predictions.
- **Continual Test-Time Adaptation (CoTTA)** (Wang et al., 2022): takes a different approach by optimizing all model parameters with an entropy objective on augmentation averaged predictions and combines it with stochastic weight restore to prevent catastrophic forgetting.

5.1. Natural Temporal Distribution Drifts

We start by evaluating the adaptation abilities of STAD to natural temporal drifts on two image classification datasets of the Wild-Time benchmark (Yao et al., 2022). While only a binary problem, the Yearbook task is difficult as the images are low-resolution (32-dimensional feature space). FMoW-Time presents an even more challenging setting: 62 classes, high-resolution images, 1024-dimensional feature space. For Yearbook, we report both the vMF and Gaussian STAD variants. For FMoW-Time, the high-dimensional representation space makes the Gaussian model too computationally costly so we evaluate just the vMF version. For Yearbook we report results for a batch size of 2048 comprising all images of a year in one batch; for FMoW, we re-use the training batch size of 64.

Table 1. Accuracy on Wild-Time benchmarks averaged over three random training seeds

Methods	Yearbook	FMoW-Time
Source Model	81.30 ± 4.18	68.94 ± 0.20
Batch Norm (BN)	84.54 ± 2.10	10.14 ± 0.04
TENT	84.53 ± 2.11	10.21 ± 0.01
CoTTA	84.35 ± 2.13	10.19 ± 0.04
STAD-vMF	85.50 ± 1.30	**86.25** ± 1.18
STAD-vMF + BN	86.20 ± 1.23	9.26 ± 1.97
STAD-Gaussian	86.22 ± 0.84	-
STAD-Gaussian + BN	**86.56** ± 1.08	-

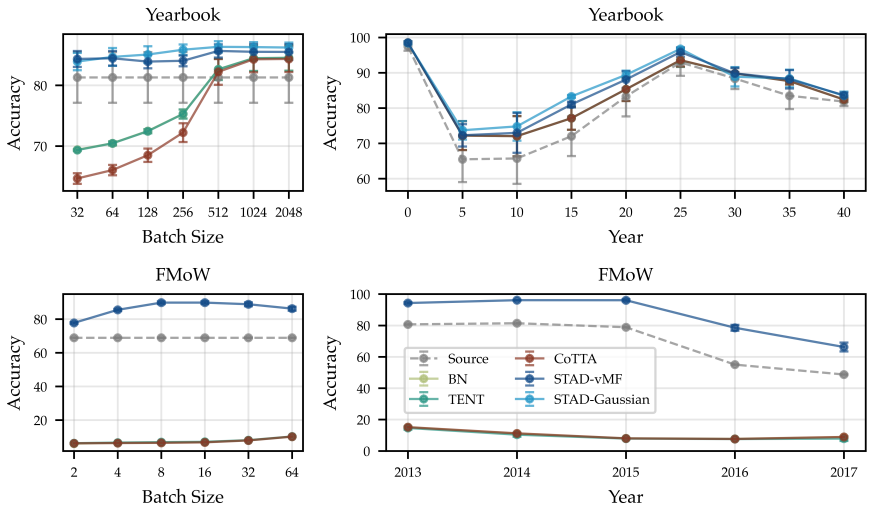

Figure 3. Adaptation performance on Wild-Time benchmarks: *First column*: Accuracy for different batch sizes. STAD is more robust to small batch sizes compared to baselines. *Right column*: Adaptation accuracy over different test time points. STAD reliably adapts to the distribution shifts. Baselines perform similarly, which is depicted by overlaying accuracy trajectories.

STAD reliably adapts to natural shifts over time Table 1 shows overall accuracy, averaged over all time steps and three random seeds. Our methods best adapt to the natural temporal shift present in both datasets, improving upon the source model in both cases. Strikingly, traditional CTTA methods collapse on FMoW-Time, falling well below the source accuracy by over 58 percentage points. On the other hand, vMF adaptation gains over 17 percentage points on the source model. This is an example of when adapting a powerful feature extractor such as the DenseNet (Huang et al., 2017) pretrained on ImageNet (Deng et al., 2009) can lead to catastrophic forgetting. On Yearbook, both Gaussian and vMF outperform baselines with the fully parameterized Gaussian model expectedly modeling the distribution shift better than the light-weight vMF model. We also test our last layer adaptation in combination with BN which modifies the feature extractor. This additionally, yields a marginal performance increase of $< 1\%$ point on Yearbook. Figure 3 *(right)* displays adaptation performance over different timestamps. While adaptation accuracy of baseline methods decays the more samples are seen on FMoW, our method keeps a constant performance gain to the source model.

STAD excels on small batch sizes Previous work (Liang et al., 2023; Yu et al., 2023; Nado et al., 2020) has observed that test-time adaptation methods are quite sensitive to the size of the test batch. To assess the impact of batch sizes, we also perform adaptation on Yearbook and FMoW using smaller batch size values. Figure 3 shows accuracy as a function of test batch size. Leveraging Bayesian principles, STAD is extremely robust to the test batch size, improving upon the source model across all batch sizes. In contrast, BN, TENT and CoTTA harm the source model when batch sizes are smaller than 512 on Yearbook and fail to improve entirely for FMoW. Adapting in small-sample environments is particularly crucial in continual domain adaptation when the task requires that predictions be made quickly.

5.2. Ablation Study: How important is modeling the dynamics?

We next investigate the importance of STAD's temporal component. STAD is proposed with the assumption that adapting the class prototypes based on those of the previous time step facilitates both rapid and reliable adaptation. However, one could also consider a static version of STAD that does not have a transition model (Equation (1)). Rather, the class prototypes are computed as a standard mixture model (Equation (2)) and without considering previously inferred prototypes. Table 2 presents the accuracy differences between the static and dynamic versions of STAD in percentage points. Removing STAD's transition component results in a *substantial performance drop* of up to 28 percentage points. This supports our assumption that state-space models are well-suited for the task of continual adaptation.

Table 2. Difference in accuracy between dynamic and static versions of STAD (i.e. when removing the transition model). Performance drops substantially on Wild-Time datasets.

Variant	Yearbook	FMoW	CIFAR-10-C
STAD-vMF w/o dynamics	–24.47	–17.38	–3.41
STAD-Gaussian w/o dynamics	–28.43	–	–

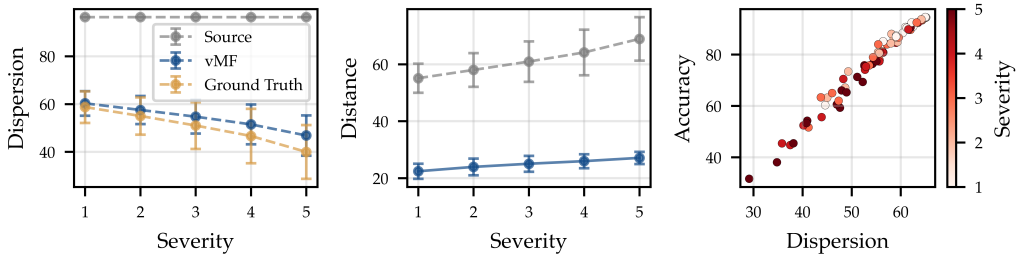

Figure 4. Analysis of cluster structure on CIFAR-10-C: *Left:* Dispersion in angular degrees increases with corruption severity, causing the ground truth cluster centers to become more similar. *Middle:* The angular distance to the ground truth cluster centers is smaller for STAD compared to the source model. *Right:* Dispersion of inferred prototypes can predict when adaptation with STAD is discouraged.

5.3. Limitation: Representations are a bottleneck

We next turn to the CIFAR-10-C dataset, which is the most commonly used benchmark in the CTTA literature. Table 3 displays adaptation accuracy averaged across all corruption types. STAD improves upon the source model for all severity levels, but does not outperform TENT or CoTTA. We suspect this reversal in performance (i.e. STAD performing better than baselines on natural but worse on synthetic shifts) is related to the quality of the source model's representations. Recall that, unlike TENT, CoTTA or BN, STAD adapts only the model's last layer. While this has benefits for STAD's computational efficiency and wide applicability, it is also a limitation in that STAD can only be as good as the last-layer representations allow.

To investigate representation quality as the cause for this performance drop, we employ the *dispersion* metric (Ming et al., 2023) developed for hyperspherical features: $\text{dis}(\mathbf{W}_t) = \frac{1}{K}\sum_{k=1}^{K}\frac{1}{K-1}\sum_{l\neq k}^{K}\mathbf{w}_{t,k}^T\mathbf{w}_{t,l}$, where $\mathbf{w}_{t,k}$ denotes the class prototypes. Dispersion measures how far apart (in angular degrees) prototypes of different classes are. High dispersion values indicate prototypes point in different directions and thus are desirable. Figure 4 *(left)* shows dispersion on CIFAR-10-C. As corruption severity increases (x-axis), the dispersion of STAD's representations (blue line) decreases. However, this is not due to any mis-estimation problem with STAD as the ground truth representations (yellow line)—computed using the true test labels—also decrease in quality. In Figure 4 *(middle)*, we confirm that STAD is appropriately adapting by measuring the angular distance to the ground-truth representations. STAD (blue line) is substantially closer to the ground truth than the source model (gray line) is, showing that STAD well tracks the class prototypes as they evolve.

Dispersion measures the quality of representation, making it a potential diagnostic tool. If dispersion is high, STAD is all that is needed for good performance. If dispersion is low, a full-model adaptation strategy such as BN is required. In Figure 4 *(right)*, we plot accuracy vs dispersion, showing that they positively correlate. This enables real-time signaling of insufficient representation quality, alerting the user to potential adaptation risks without the need for labels. Recognizing when adaptation fails is of critical importance in practice. Recent work has shown that most CTTA methods eventually collapse (Press et al., 2024a;b), but understanding why and when this phase transition occurs is an open problem.

Table 3. Accuracy on CIFAR-10-C for different severity levels averaged over all corruption types

Method	Corruption severity 1	2	3	4	5	Mean
Source	86.90	81.34	74.92	67.64	56.48	73.46
BN	90.18	88.16	86.24	83.18	79.27	85.41
TENT	**90.87**	**89.70**	88.32	85.89	83.09	87.57
CoTTA	90.62	89.42	**88.55**	**87.28**	**85.27**	**88.23**
STAD-vMF	88.21	83.68	78.42	72.19	62.44	76.99
STAD-vMF + BN	90.16	88.11	86.24	83.17	79.33	85.40

6. Conclusion

We have presented a novel test-time adaptation strategy, *State-space Test-time ADaptation* (STAD), based on a probabilistic state-space model. Both our Gaussian and vMF variants of STAD track how the last-layer evolves under distribution shift, allowing a deployed model to perform unsupervised adaptation to the shift. Our framework outperforms state-of-the-art competitors such as TENT and CoTTA on Wild-Time benchmarks. Yet we also identify points of improvement, as we found that random corruptions applied to CIFAR-10 degraded the representations to a degree that our method could be competitive with but not outperform these baselines adapting the model backbone. For future work, we will study diagnostics that reliably identify when the last-layer representations are suitable, more intensive adaptation is needed or if adaptation is possible at all. We will also investigate non-linear models of shift dynamics.

References

A. F. Ansari, A. Heng, A. Lim, and H. Soh. Neural continuous-discrete state space models for irregularly-sampled time series. In *International Conference on Machine Learning*, pages 926–951. PMLR, 2023.

H. Bai, Y. Ming, J. Katz-Samuels, and Y. Li. Hypo: Hyperspherical out-of-distribution generalization. *International Conference on Learning Representations*, 2024.

A. Banerjee, I. S. Dhillon, J. Ghosh, S. Sra, and G. Ridgeway. Clustering on the unit hypersphere using von mises-fisher distributions. *Journal of Machine Learning Research*, 6(9), 2005.

P. Becker, H. Pandya, G. Gebhardt, C. Zhao, C. J. Taylor, and G. Neumann. Recurrent kalman networks: Factorized inference in high-dimensional deep feature spaces. In *International Conference on Machine Learning*, pages 544–552. PMLR, 2019.

C. M. Bishop and N. M. Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

A. Calabrese and L. Paninski. Kalman filter mixture model for spike sorting of non-stationary data. *Journal of neuroscience methods*, 196(1):159–169, 2011.

P. G. Chang, G. Durán-Martín, A. Shestopaloff, M. Jones, and K. P. Murphy. Low-rank extended kalman filtering for online learning of neural networks from streaming data. In *Proceedings of The 2nd Conference on Lifelong Learning Agents*, volume 232, pages 1025–1071. PMLR, 2023.

G. Christie, N. Fendley, J. Wilson, and R. Mukherjee. Functional map of the world. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6172–6180, 2018.

F. Croce, M. Andriushchenko, V. Sehwag, E. Debenedetti, N. Flammarion, M. Chiang, P. Mittal, and M. Hein. Robustbench: a standardized adversarial robustness benchmark. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1*, 2021.

M. De Lange, R. Aljundi, M. Masana, S. Parisot, X. Jia, A. Leonardis, G. Slabaugh, and T. Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(7):3366–3385, 2021.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

M. Döbler, R. A. Marsden, and B. Yang. Robust mean teacher for continual and gradual test-time adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7704–7714, 2023.

M. Fraccaro, S. D. Kamronn, U. Paquet, and O. Winther. A disentangled recognition and nonlinear dynamics model for unsupervised learning. *Advances in Neural Information Processing Systems*, 2017.

S. Ginosar, K. Rakelly, S. Sachs, B. Yin, and A. 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.

T. Gong, J. Jeong, T. Kim, Y. Kim, J. Shin, and S.-J. Lee. Note: Robust continual test-time adaptation against temporal correlation. *Advances in Neural Information Processing Systems*, 2022.

S. Gopal and Y. Yang. Von mises-fisher clustering models. In *International Conference on Machine Learning*, pages 154–162. PMLR, 2014.

D. Hendrycks and T. Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *International Conference on Learning Representations*, 2019.

G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4700–4708, 2017.

R. E. Kalman. A new approach to linear filtering and prediction problems. 1960.

M. Karl, M. Soelch, J. Bayer, and P. van der Smagt. Deep variational bayes filters: Unsupervised learning of state space models from raw data. *International Conference on Learning Representations*, 2017.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 2015.

P. W. Koh, S. Sagawa, H. Marklund, S. M. Xie, M. Zhang, A. Balsubramani, W. Hu, M. Yasunaga, R. L. Phillips, I. Gao, et al. Wilds: A benchmark of in-the-wild distribution shifts. In *International Conference on Machine Learning*, pages 5637–5664. PMLR, 2021.

R. G. Krishnan, U. Shalit, and D. Sontag. Deep kalman filters. *arXiv preprint (arXiv:1511.05121)*, 2015.

J. Liang, R. He, and T. Tan. A comprehensive survey on test-time adaptation under distribution shifts. *arXiv preprint (arXiv:2303.15361)*, 2023.

K. V. Mardia and P. E. Jupp. *Directional statistics*. John Wiley & Sons, 2009.

P. Mettes, E. Van der Pol, and C. Snoek. Hyperspherical prototype networks. *Advances in Neural Information Processing Systems*, 2019.

Y. Ming, Y. Sun, O. Dia, and Y. Li. How to exploit hyperspherical embeddings for out-of-distribution detection? *International Conference on Learning Representations*, 2023.

Z. Nado, S. Padhy, D. Sculley, A. D'Amour, B. Lakshminarayanan, and J. Snoek. Evaluating prediction-time batch normalization for robustness under covariate shift. *arXiv preprint (arXiv:2006.10963)*, 2020.

S. Niu, J. Wu, Y. Zhang, Y. Chen, S. Zheng, P. Zhao, and M. Tan. Efficient test-time model adaptation without forgetting. In *International Conference on Machine Learning*, pages 16888–16905. PMLR, 2022.

V. M. Patel, R. Gopalan, R. Li, and R. Chellappa. Visual domain adaptation: A survey of recent advances. *IEEE signal processing magazine*, 32(3):53–69, 2015.

O. Press, S. Schneider, M. Kümmerer, and M. Bethge. Rdumb: A simple approach that questions our progress in continual test-time adaptation. *Advances in Neural Information Processing Systems*, 2024a.

O. Press, R. Shwartz-Ziv, Y. LeCun, and M. Bethge. The entropy enigma: Success and failure of entropy minimization. *arXiv preprint (arXiv:2405.05012)*, 2024b.

M. Schirmer, M. Eltayeb, S. Lessmann, and M. Rudolph. Modeling irregular time series with continuous recurrent units. In *International Conference on Machine Learning*, pages 19388–19405. PMLR, 2022.

S. Schneider, E. Rusak, L. Eck, O. Bringmann, W. Brendel, and M. Bethge. Improving robustness against common corruptions by covariate shift adaptation. *Advances in Neural Information Processing Systems*, 2020.

J. Song, J. Lee, I. S. Kweon, and S. Choi. Ecotta: Memory-efficient continual test-time adaptation via self-distilled regularization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11920–11929, 2023.

M. K. Titsias, A. Galashov, A. Rannen-Triki, R. Pascanu, Y. W. Teh, and J. Bornschein. Kalman filter for online classification of non-stationary data. *arxiv preprint (arXiv:2306.08448)*, 2023.

D. Wang, E. Shelhamer, S. Liu, B. A. Olshausen, and T. Darrell. Tent: Fully test-time adaptation by entropy minimization. *International Conference on Learning Representations*, 2021.

Q. Wang, O. Fink, L. Van Gool, and D. Dai. Continual test-time domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7201–7211, 2022.

G. Wilson and D. J. Cook. A survey of unsupervised deep domain adaptation. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 11(5):1–46, 2020.

H. Yao, C. Choi, B. Cao, Y. Lee, P. W. W. Koh, and C. Finn. Wild-time: A benchmark of in-the-wild distribution shift over time. *Advances in Neural Information Processing Systems*, 2022.

Y. Yu, L. Sheng, R. He, and J. Liang. Benchmarking test-time adaptation against distribution shifts in image classification. *arXiv preprint (arXiv:2307.03133)*, 2023.

S. Zagoruyko and N. Komodakis. Wide residual networks. *arXiv preprint (arXiv:1605.07146)*, 2016.

H. Zhu, C. Balsells-Rodas, and Y. Li. Markovian gaussian process variational autoencoders. In *International Conference on Machine Learning*, pages 42938–42961. PMLR, 2023.

A. STAD-Gaussian

We use a linear Gaussian transition model to describe the weight evolution over time: For each class k, the weight vector evolves according to a linear drift parameterized by a class-specific transition matrix $\boldsymbol{A}_k \in \mathbb{R}^{D\times D}$. This allows each class to have independent dynamics. The transition noise follows a multivariate Gaussian distribution with zero mean and global covariance $\boldsymbol{\Sigma}^{\text{trans}} \in \mathbb{R}^{D\times D}$. The transition noise covariance matrix is a shared parameter across classes and time points to prevent overfitting and keep parameter size at bay. Equation (11) states the Gaussian transition density.

$$\text{Transition model:} \quad p(\mathbf{W}_t|\mathbf{W}_{t-1}) = \prod_{k=1}^{K} \mathcal{N}(\mathbf{w}_{t,k}|\boldsymbol{A}_k\mathbf{w}_{t-1,k}, \boldsymbol{\Sigma}^{\text{trans}}) \tag{11}$$

$$\text{Emission model:} \quad p(\mathbf{H}_t|\mathbf{W}_t) = \prod_{n=1}^{N_t}\sum_{k=1}^{K} \pi_{t,k}\mathcal{N}(\mathbf{h}_{t,n}|\mathbf{w}_{t,k}, \boldsymbol{\Sigma}^{\text{ems}}) \tag{12}$$

Equation (12) gives the emission model of the observed features $\mathbf{H}_t$ at time t. As in Equation (2), the features at a given time t are generated by a mixture distribution with mixing coefficient $\pi_{t,k}$. The emission density of each of the K component is a multivariate normal with the weight vector of class k at time t as mean and $\boldsymbol{\Sigma}^{\text{ems}} \in \mathbb{R}^{D\times D}$ as class-independent covariance matrix. The resulting model can be seen as a mixture of K Kalman filters. Variants of it has found application in applied statistics (Calabrese and Paninski, 2011).

Posterior inference We use the EM objective of Equation (4) to maximize for the model parameters $\phi = \{\{\boldsymbol{A}_k, \{\pi_{t,k}\}_{t=1}^{T}\}_{k=1}^{K}, \boldsymbol{\Sigma}^{\text{trans}}, \boldsymbol{\Sigma}^{\text{ems}}\}$. Thanks to the linearity and Gaussian assumptions, the posterior expectation $\mathbb{E}_{p(\mathbf{W},\mathbf{C}|\mathbf{H})}[\cdot]$ in Equation (4) can be computed analytically using the well known Kalman filter predict, update and smoothing equations (Calabrese and Paninski, 2011; Bishop and Nasrabadi, 2006).

Complexity The closed form computations of the posterior $p(\mathbf{W}_t|\mathbf{H}_{1:t})$ and smoothing $p(\mathbf{W}_t|\mathbf{H}_{1:T})$ densities come at a cost as they involve amongst others matrix inversions of dimensionality $D \times D$. This results in considerable computational costs and can lead to numerical instabilities when feature dimension D is large. In addition, the parameter size scales $K \times D^2$ risking overfitting and consuming substantial memory. These are limitations of the Gaussian formulation making it costly for high-dimensional feature spaces and impractical in low resource environments requiring instant predictions.

B. Inference for STAD-vMF

Complete-data log likelihood Using the von Mises-Fisher distribution as hyperspherical transition (Equation (7)) and emission model (Equation (8)), the log of the complete-data likelihood in Equation (3) becomes

$$\log p(\mathbf{H}_{1:T}, \mathbf{W}_{1:T}, \mathbf{C}_{1:T}) = \sum_{k}^{K} \log p(\mathbf{w}_{1,k}) \tag{13}$$

$$+ \sum_{t=1}^{T} \sum_{n=1}^{N_t} \log p(\mathbf{c}_{t,n}) + \sum_{k=1}^{K} \mathbf{c}_{t,n,k} \log p(\mathbf{h}_{t,n} | \mathbf{w}_{t,k}, \kappa^{\text{ems}}) \tag{14}$$

$$+ \sum_{t=2}^{T} \sum_{k=1}^{K} \log p(\mathbf{w}_{t,k} | \mathbf{w}_{t-1,k}, \kappa^{\text{trans}}) \tag{15}$$

$$= \sum_{k}^{K} \log C_D(\kappa_{0,k}) + \kappa_{0,k} \boldsymbol{\mu}_{0,k}^T \mathbf{w}_{1,k} \tag{16}$$

$$+ \sum_{t=1}^{T} \sum_{n=1}^{N_t} \sum_{k=1}^{K} c_{n,t,k} \big(\log \pi_{t,k} + \log C_D(\kappa^{\text{ems}}) + \kappa^{\text{ems}} \mathbf{w}_{t,k}^T \mathbf{h}_{t,n} \big) \tag{17}$$

$$+ \sum_{t=2}^{T} \sum_{k=1}^{K} \log C_D(\kappa^{\text{trans}}) + \kappa^{\text{trans}} \mathbf{w}_{t-1,k}^T \mathbf{w}_{t,k} \tag{18}$$

where $\kappa_{0,k}$ and $\boldsymbol{\mu}_{0,k}$ denote the parameters of the first time step. In practise, we set $\boldsymbol{\mu}_{0,k}$ to the source weights and $\kappa_{0,k} = 100$.

Variational EM objective As described in Section 3.3, we approximate the posterior $p(\mathbf{W}_{1:T}, \mathbf{C}_{1:T} | \mathbf{H}_{1:T})$ with a variational distribution $q(\mathbf{W}_{1:T}, \mathbf{C}_{1:T})$ assuming the factorised form

$$q(\mathbf{W}_{1:T}, \mathbf{C}_{1:T}) = \prod_{t=1}^{T} \prod_{k=1}^{K} q(\mathbf{w}_{t,k}) \prod_{n=1}^{N_t} q(\mathbf{c}_{n,t}), \tag{19}$$

where we parameterise $q(\mathbf{w}_{t,k})$ and $q(\mathbf{c}_{n,t})$ with

$$q(\mathbf{w}_{t,k}) = \texttt{vMF}(\,\cdot\,; \boldsymbol{\rho}_{\boldsymbol{t},\boldsymbol{k}}, \gamma_{t,k}) \quad q(\mathbf{c}_{n,t}) = \texttt{Cat}(\,\cdot\,; \boldsymbol{\lambda}_{\boldsymbol{n},\boldsymbol{t}}) \quad \forall t, n, k. \tag{20}$$

We obtain the variational EM objective

$$\arg\max_{\phi} \mathbb{E}_q \big[\log p(\mathbf{H}_{1:T}, \mathbf{W}_{1:T}, \mathbf{C}_{1:T}) \big], \tag{21}$$

where $\mathbb{E}_{q(\mathbf{W}_{1:T}, \mathbf{C}_{1:T})}$ is denoted $\mathbb{E}_q$ to reduce clutter.

E-step Taking the expectation of the complete-data log likelihood (Equation (13)) with respect to the variational distribution (Equation (19)) gives

$$\mathbb{E}_q[\log p(\mathbf{H}_{1:T}, \mathbf{W}_{1:T}, \mathbf{C}_{1:T})] = \sum_{k}^{K} \log C_D(\kappa_{0,k}) + \kappa_{0,k} \boldsymbol{\mu}_{0,k}^T \mathbb{E}_q[\mathbf{w}_{1,k}] \tag{22}$$

$$+ \sum_{t=1}^{T} \sum_{n=1}^{N_t} \sum_{k=1}^{K} \mathbb{E}_q[c_{n,t,k}] \big(\log \pi_{t,k} + \log C_D(\kappa^{\text{ems}}) + \kappa^{\text{ems}} \mathbb{E}_q[\mathbf{w}_{t,k}]^T \mathbf{h}_{t,n} \big) \tag{23}$$

$$+ \sum_{t=2}^{T} \sum_{k=1}^{K} \log C_D(\kappa^{\text{trans}}) + \kappa^{\text{trans}} \mathbb{E}_q[\mathbf{w}_{t-1,k}]^T \mathbb{E}_q[\mathbf{w}_{t,k}] \tag{24}$$

Solving for the variational parameters, we obtain

$$\lambda_{n,t,k} = \frac{\beta_{n,t,k}}{\sum_{j=1}^{K} \beta_{n,t,j}} \quad \text{with} \quad \beta_{n,t,k} = \pi_{t,k} C_D(\kappa^{\text{ems}}) \exp(\kappa^{\text{ems}} \mathbb{E}_q[\mathbf{w}_{t,k}]^T \mathbf{h}_{n,t}) \tag{25}$$

$$\boldsymbol{\rho}_{t,k} = \frac{\kappa^{\text{trans}} \mathbb{E}_q[\mathbf{w}_{t-1,k}] + \kappa^{\text{ems}} \sum_{n=1}^{N_t} \mathbb{E}_q[c_{n,t,k}] \mathbf{h}_{n,t} + \kappa^{\text{trans}} \mathbb{E}_q[\mathbf{w}_{t+1,k}]}{\gamma_{t,k}} \tag{26}$$

$$\gamma_{t,k} = ||\boldsymbol{\rho}_{t,k}|| \tag{27}$$

The expectations are given by

$$\mathbb{E}[c_{n,t,k}] = \lambda_{n,t,k} \tag{28}$$

$$\mathbb{E}[\mathbf{w}_{t,k}] = A_D(\gamma_{t,k}) \boldsymbol{\rho_{t,k}}, \tag{29}$$

where $A_D(\kappa) = \frac{I_{D/2}(\kappa)}{I_{D/2-1}(\kappa)}$ and $I_v(a)$ denotes the modified Bessel function of the first kind with order v and argument a.

M-step Maximizing objective (Equation (21)) with respect to the model parameters $\phi = \{\kappa^{\text{trans}}, \kappa^{\text{ems}}, \{\pi_{t,k}\}_{t=1}^{T}\}_{k=1}^{K}\}$ gives

$$\hat{\kappa}^{\text{trans}} = \frac{\bar{r}^{\text{trans}} D - (\bar{r}^{\text{trans}})^3}{1 - (\bar{r}^{\text{trans}})^2} \quad \text{with} \quad \bar{r}^{\text{trans}} = \left\| \frac{\sum_{t=2}^{T} \sum_{k=1}^{K} \mathbb{E}_q[\mathbf{w}_{t-1,k}]^T \mathbb{E}_q[\mathbf{w}_{t,k}]}{(T-1) \times K} \right\| \tag{30}$$

$$\hat{\kappa}^{\text{ems}} = \frac{\bar{r}^{\text{ems}} D - (\bar{r}^{\text{ems}})^3}{1 - (\bar{r}^{\text{ems}})^2} \quad \text{with} \quad \bar{r}^{\text{ems}} = \left\| \frac{\sum_{t=2}^{T} \sum_{k=1}^{K} \sum_{n=1}^{N_t} \mathbb{E}_q[\mathbf{c}_{n,t,k}] \mathbb{E}_q[\mathbf{w}_{t,k}]^T \mathbf{h}_{n,t}}{\sum_{t=1}^{T} N_t} \right\| \tag{31}$$

$$\pi_{t,k} = \frac{\sum_{n=1}^{N_t} \mathbb{E}[c_{n,t,k}]}{N_t} \tag{32}$$

Here we made use of the approximation from Banerjee et al. (2005) to compute an estimate for κ,

$$\hat{\kappa} = \frac{\bar{r} D - \bar{r}^3}{1 - \bar{r}^2} \quad \text{with} \quad \bar{r} = A_D(\hat{\kappa}). \tag{33}$$

C. Experimental Details

We next list details on the experimental setup and hyperparameter configurations. All experiments are performed on NVIDIA RTX 6000 Ada with 48GB memory.

C.1. Yearbook and FMoW

We follow the Eval-Fix protocol of Wild-Time to assess adaptation performance on Yearbook and FMoW. In Eval-Fix, source models are trained on a fixed time period and evaluated on unseen, future time points. We use the models trained by Yao et al. (2022) on the training time period and evaluate them on the test period. Yao et al. (2022) uses a range of different training procedures. We use the checkpoints for plain empirical risk minimization. Three different random training seeds are provided by Yao et al. (2022). All adaptation methods are continuously running without resets. Each method uses one optimization step (via entropy minimization for TENT and CoTTA and Expectation-Maximization for STAD).

By the nature of test-time adaptation, choosing hyperparameters is difficult since one cannot assume access to a validation set of the test distribution in practise. To ensure we report the optimal performance of baseline models, we conduct a grid search on the test set for relevant hyperparameters and report the performance of the best setting. The hyperparameters tested as well as other configurations are listed next.

BN (Schneider et al., 2020; Nado et al., 2020) Normalization statistics during test-time adaptation are a running estimates of both the training data and the incoming test statistics. No hyperparameter optimization is necessary here.

TENT (Wang et al., 2021) Like in BN, the normalization statistics are based on both training and test set. As in Wang et al. (2021), we use the same optimizer settings for test-time adaptation as used for training, except for the learning rate that we find via grid search on $\{1e^{-3}, 1e^{-4}, 1e^{-5}, 1e^{-6}, 1e^{-7}\}$. For both yearbook and FMoW, Adam optimizer (Kingma and Ba, 2015) is used.

CoTTA (Wang et al., 2022) We use the same optimizer as used during training (Adam optimizer (Kingma and Ba, 2015)). For hyperparameter optimization we follow the parameter suggestions by Wang et al. (2022) and conduct a grid search for the learning rate ($\{1e^{-3}, 1e^{-4}, 1e^{-5}, 1e^{-6}, 1e^{-7}\}$), EMA factor ($\{0.99, 0.999, 0.9999\}$) and restoration factor ($\{0, 0.001, 0.01, 0.1\}$). We follow (Wang et al., 2022) by determining the augmentation confidence threshold as a function of the 5 % percentile for the softmax prediction confidence on the source images from the source model. Note that this requires access to the source data.

STAD-Gaussian We initialize the mixing coefficients with $\pi_{t,k} = \frac{1}{K} \forall t, k$, the transition covariance matrix with $\mathbf{\Sigma}^{\text{trans}} = 0.01 \times \mathbf{I}$ and the emission covariance matrix with $\mathbf{\Sigma}^{\text{ems}} = 0.5 \times \mathbf{I}$. The prototypes at time $t = 1$ are initialized with the source weights. We found a normalization of the representations to be also beneficial for STAD-Gaussian. Note that despite normalization, the two models are not equivalent. STAD-Gaussian models the correlation between different dimensions of the representations and is therefore more expressive, while STAD-vMF assumes an isotropic variance.

STAD-vMF We initialize the mixing coefficients with $\pi_{t,k} = \frac{1}{K} \forall t, k$ and the prototypes at time $t = 1$ with the source weights. For yearbook, we employ class specific noise parameters initialized with $\kappa_k^{\text{trans}} = 100$ and $\kappa_k^{\text{ems}} = 100$. For FMoW, we found a more restricted transition noise model beneficial. We follow suggestions by Gopal and Yang (2014) to keep noise concentration parameters fixed instead of learning them via maximum likelihood in order to maintain a regularization term. We thus keep the noise parameters fixed at $\kappa^{\text{trans}} = 1000$ and $\kappa^{\text{ems}} = 100$.

C.2. CIFAR-10-C

For the experiments on CIFAR-10-C, we construct a gradual distribution shift setting by increasing the corruption severity sequentially from level 1 to level 5. We adapt each model separately for each of the 15 corruption types. The source model is a WideResNet-28 (Zagoruyko and Komodakis, 2016) from RobustBench (Croce et al., 2021). CIFAR-10-C is a well studied benchmark in CTTA and thus we take the hyperparameter settings of baseline methods reported in previous work (Wang et al., 2022).

BN As in Appendix C.1, normalization statistics during test-time adaptation are a running estimates of both the training statistics and the incoming test statistics. No hyperparameter optimization is required.

TENT As in (Wang et al., 2022), we use Adam optimizer with learning rate 1e-3.

CoTTA We follow (Wang et al., 2022) and use Adam optimizer with learning rate 1e-3. The EMA factor is set to 0.999, the restoration factor is 0.01 and the augmentation confidence threshold is 0.92.

STAD-vMF We initialize the mixing coefficients with $\pi_{t,k} = \frac{1}{K} \forall t, k$, the transition concentration parameter with $\kappa^{\text{trans}} = 100$, and the emission concentration parameter with $\kappa^{\text{ems}} = 100$. The prototypes at time $t = 1$ are initialized with the source weights.