Protected Test-Time Adaptation via Online Entropy Matching: A Betting Approach

Yarin Bar[1]* **Shalev Shaer**[2]* **Yaniv Romano**[1,2]

[1]Department of Computer Science, Technion—Israel Institute of Technology
[2]Department of Electrical and Computer Engineering, Technion—Israel Institute of Technology
{yarinbar,shalev.shaer}@campus.technion.ac.il
yromano@technion.ac.il

Abstract

We present a novel approach for test-time adaptation via online self-training, consisting of two components. First, we introduce a statistical framework that detects distribution shifts in the classifier's entropy values obtained on a stream of unlabeled samples. Second, we devise an online adaptation mechanism that utilizes the evidence of distribution shifts captured by the detection tool to dynamically update the classifier's parameters. The resulting adaptation process drives the distribution of test entropy values obtained from the self-trained classifier to match those of the source domain, building invariance to distribution shifts. This approach departs from the conventional self-training method, which focuses on minimizing the classifier's entropy. Our approach combines concepts in betting martingales and online learning to form a detection tool capable of quickly reacting to distribution shifts. We then reveal a tight relation between our adaptation scheme and optimal transport, which forms the basis of our novel self-supervised loss. Experimental results demonstrate that our approach improves test-time accuracy under distribution shifts while maintaining accuracy and calibration in their absence, outperforming leading entropy minimization methods across various scenarios.

1 Introduction

The deployment of machine learning (ML) models in real-world settings presents a significant challenge, as these models often encounter testing environments (target domains) that differ from their training, source domain [1–15]. Consider, for example, an image recognition system employed for medical diagnostic support [16–20], where the quality of images acquired during testing deviates from the training data due to factors such as equipment degradation and novel illumination conditions. ML models are sensitive to such distribution shifts, often resulting in performance deterioration, which can be unexpected [15]. Ultimately, we want predictive models to dynamically adapt to new testing environments without the laborious work required to annotate new, up-to-date labels.

Recognizing this pressing need, there has been a surge in the development of adaptation methodologies to enhance test-time robustness to shifting distributions [21–28]. One commonly used approach involves jointly training a model on both the source and target domains [29–34]. However, such train-time adaptation methods assume access to unlabeled test data from the target domains, limiting the ability to adapt the model to new domains that emerge during testing. To overcome this limitation, test-time adaptation techniques offer strategies that dynamically update the model parameters as new unlabeled test points become available. In particular, leading methodologies draw inspiration from

*[1]Equal Contribution.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

the relationship between the entropy of estimated class probabilities—a measure of confidence—and model accuracy [35–43]. Empirical evidence highlights that lower entropy often corresponds to higher accuracy, encouraging the development of self-supervised learning approaches that adjust model parameters by minimizing the entropy loss or the cross-entropy through the assignment of pseudo- or soft-labels to test points [44–48].

While test-time adaptation techniques have shown promise in enhancing test accuracy under domain shifts, there is a caveat: minimizing entropy or related self-supervised loss functions *without control* can lead to overconfident predictions, and may suffer from undesired, noisy model updates [49–52]. In extreme cases, this approach may even cause the model to collapse and produce trivial predictions [53, 54]. Indeed, without careful implementation and tuning, these techniques may not improve—or could even reduce [34]—the predictive performance in realistic settings.

In this paper, we present a novel, statistically principled approach to test-time adaptation via self-training. Our methodology is built upon two key pillars. First, we introduce an online statistical framework that monitors and detects distribution shifts in the test data influencing the models' predictions. We achieve this by sequentially testing whether the distribution of the classifier's entropy values obtained during testing deviates from the ones corresponding to the source domain. Armed with this monitoring tool, we then devise an online adaptation mechanism that leverages the accumulated evidence of distribution shifts to adaptively update model parameters. This mechanism drives the distribution of the self-trained classifier's entropy values, obtained on test data, to closely match the distribution of entropy values when applying the original model to the source domain. As a result, our proposed **Protected Online Entropy Matching** (`POEM`) method adapts the model on the fly in a controlled manner: in the absence of a distribution shift, our approach has a "no-harm" effect both on accuracy and calibration of the model, whereas under distribution shifts our experiments demonstrate an improvement of the test time accuracy, often surpassing state-of-the-art methods.

Contributions

(i) We present a sequential test for classification entropy drift detection, building on betting martingales [55–57] and online learning optimization [58–60] to provably attain fast reactions to shifting data. (ii) Inspired by [61], we show how to utilize the test martingale to analytically design a mapping function that transports the classifier entropies obtained at test time to resemble those of the source domain. Under certain assumptions, we establish connections between our online testing approach and optimal transport [62] as a mechanism for distribution matching. (iii) This observation sets the foundation of the entropy-matching loss function used in `POEM`. (iv) Numerical experiments in both continual and single-shift settings demonstrate that our approach is competitive and often outperforms strong benchmark methods that build on entropy minimization. These experiments are conducted using commonly used predictive models (ViT [63] and ResNet [64]) on standard benchmark datasets: ImageNet-C [65], CIFAR10-C, CIFAR100-C, and OfficeHome [66]. A software package that implements our methods is available at `https://github.com/yarinbar/poem`.

2 Preliminaries

2.1 Problem setup

To formalize the problem, consider a K-class classification problem with labeled training data $(X_i^s, Y_i^s)_{i=1}^n$ from a source domain, sampled i.i.d. from the source distribution P_{XY}^s. Here, $X^s \in \mathbb{R}^d$ represents observed covariates and $Y^s \in \{1, \dots, K\}$ is the corresponding label. During testing, we encounter a stream of points X_j^t with unknown labels Y_j^t, sampled from an unknown target distribution P_{XY}^j that may shift over time $j = 1, 2, \dots$. To define the shifting mechanism, let $T_j : \mathbb{R}^d \to \mathbb{R}^d$ be an unknown corruption/shift function, resulting in test instances $X_j^t = T_j(X^s)$ with X^s being a fresh sample from P^s. For instance, in datasets such as ImageNet-C, corruptions involve modifications such as blur or changes in illumination applied to clean source images. While such transformations alter the marginal P_X^j distribution of the target domain, we assume that the conditional distributions of $P_{Y|X}^s(Y^s \mid X^s)$ and $P_{Y|X}^j(Y_j^t | T_j(X^s))$ are the same for all $j = 1, 2, \dots$ [67].

2.2 Related work: test-time adaptation via self-training

Given a pre-trained classifier $f_{\hat{\theta}}$ trained on the source domain, leading test-time adaptation approaches build on the idea of self-training to update the model parameters sequentially. Denote the adapted classifier by $f_{\hat{\theta}+\omega}$, where ω represents the modification to the parameters of the original model $\hat{\theta}$ obtained by self-training during testing. The adaptation process is typically initialized with $j = 1$ and $\omega_1 = \mathbf{0}$, and involves the following set of steps:

1. Observe a fresh test point X_j^t and predict its unknown label using $f_{\hat{\theta}+\omega_j}(X_j^t)$.
2. Update the model parameters in a direction that reduces the self-supervised loss, i.e.,

$$\omega_{j+1} \leftarrow \omega_j - \eta \nabla_\omega \ell^{\text{self}} \left(f_{\hat{\theta}+\omega_j}(X_j^t) \right),$$

   where the hyper-parameter η is the step size.
3. Set $j \leftarrow j + 1$ and return to step 1.

A common choice for ℓ^{self} in test-time adaptation methods is the entropy loss:

$$\ell^{\text{ent}} \left(f_{\hat{\theta}+\omega}(x) \right) = -\sum_{y=1}^{K} f_{\hat{\theta}+\omega}(x)_y \log(f_{\hat{\theta}+\omega}(x)_y), \tag{1}$$

where $f_{\hat{\theta}+\omega}(x)_y$ is the y-th entry of the classifier's softmax layer; we omit the index j of ω for clarity.

While entropy minimization has been shown to enhance test-time robustness [36–43], it is also prone to instabilities [49–51, 68]. For instance, enforcing $f_{\hat{\theta}+\omega}(x)_y = 1$ for a fixed entry y minimizes ℓ^{ent}, but it collapses the classifier to make trivial predictions. To alleviate this, various strategies have been proposed. For example, one approach is to avoid training on samples with high entropy [42], as high entropy often correlates with erroneous pseudo labels. Another example is the utilization of a fallback mechanism that resets the adapted model back to the original model $f_{\hat{\theta}}$ when the average entropy becomes too small [43]. A more detailed review of test-time adaptation methods is given in Appendix A. Importantly, this line of work underscores the limitations of entropy minimization and highlights the need to better control its effect. This aligns with the goal of our work, which offers a distinct, statistically grounded approach for test-time adaptation. While we also build on the classifier's entropy to form the adaptation mechanism, we could integrate any alternative self-supervised loss in our online distributional matching scheme.

2.3 Testing by betting

A key component of our method is the proposal of an online test for distribution drift. The design of this test follows the framework of *testing-by-betting* [69]. Intuitively, one can interpret this testing framework as participating in a fair game. We begin with initial toy money, and at each time step, we observe a new test point and place a bet against the null hypothesis we aim to test. If the bet turns out to be correct, our wealth increases by the money we risked in the bet; otherwise, we lose, and our wealth decreases accordingly. Mathematically, the wealth process is formulated as a non-negative martingale, where a successful betting scheme is reflected in a growing martingale (wealth) trajectory, offering progressively stronger statistical evidence against the null hypothesis. However, if the null hypothesis is true, the game must be fair in the sense that it is unlikely to significantly grow our initial capital, no matter how sophisticated our betting strategy may be. This implies that, under the null hypothesis, it is unlikely that the martingale will grow significantly beyond its initial value.

The testing-by-betting framework is widely used in sequential settings. Notable examples include: one and two sample tests [70, 71], independence and conditional independence tests [71–73], exchangeability tests [56, 74–78], and more [69, 79–92]. This framework is also used for change-point detection and testing for uniformity [57, 93–95], related to our drift detection problem. We draw inspiration from the protected probabilistic regression approach [61] that combines the probability integral transform and betting martingales to improve the robustness of a cumulative distribution function (CDF) estimator to distribution shift in the data. The experiments in [61] illustrate this method's ability to enhance the accuracy of a regression model, where this protection scheme assumes access to new up-to-date labels. In contrast, we focus on a completely different setup where the labels of the test points are unknown, showing how the protected regression approach can be generalized to form a self-supervised loss function. In turn, we introduce two key contributions to test-time

adaptation via testing-by-betting. First, we present an adaptive online learning technique to optimize the betting mechanism. Second, we pioneer the application of testing-by-betting in this domain.

3 Proposed method: protected online entropy matching (POEM)

3.1 Preview of our method

Let the random variable $Z^s = \ell^{\text{ent}}(f_{\hat{\theta}}(X^s))$ be the entropy value of the original classifier applied to a fresh sample X^s from the source domain. We refer to this variable as the *source entropy*. In addition, denote by $Z_j^t = \ell^{\text{ent}}(f_{\hat{\theta}+\omega_j}(X_j^t))$, $j = 1, 2, \ldots$, a sequence of entropy values generated by the updated model, evaluated on a stream of unlabeled test data. We refer to Z^t as the *target entropy*. Our proposal uses the source and target entropies both to detect distribution shifts and adapt the model to new testing environments without relying on up-to-date labeled data. The rationale behind our method is as follows: when test data is sampled from the source distribution, there will be no deviation between the source and target entropies, implying that there is no need to adapt the model. However, statistical deviations between the source entropies Z^s and target entropies Z^t can indicate that the model encounters test data different from the training distribution. This motivates us to introduce a self-training framework that encourages the model to generate test-time entropies Z^t that closely resemble the source entropies Z^s to build invariance to shifting data.

To achieve this goal, we utilize the testing-by-betting approach and formulate our adaptation scheme as a game, in which we start with initial toy money and proceed as follows.

1. Place a bet against the null hypothesis that the unknown classifier entropy Z_j^t of the upcoming test point X_j^t will follow the same distribution as the source entropies Z^s.
2. Observe the test point X_j^t, predict its label using the model $f_{\hat{\theta}+\omega_j}$, and compute the classifier's entropy $Z_j^t = \ell^{\text{ent}}(f_{\hat{\theta}+\omega_j}(X_j^t))$.
3. Given Z_j^t, reveal the outcome of the bet using a betting function. If the bet is successful, increase the accumulated wealth; otherwise, decrease it.
4. Leverage the betting function to obtain an adapted pseudo-entropy value $\tilde{Z}_j$ that better matches the distribution of Z^s. The intuition here is that we derive $\tilde{Z}_j$ in a way that would reduce the toy money we would have gained if we had used the same betting strategy on $\tilde{Z}_j$.
5. Update the model parameters: obtain ω_{j+1} by taking a gradient step that reduces the self-supervised matching loss:[2]

$$\ell^{\text{match}}(Z_j^t, \tilde{Z}_j) = \frac{1}{2}(\ell^{\text{ent}}(f_{\hat{\theta}+\omega_j}(X_j^t)) - \tilde{Z}_j)^2. \tag{2}$$

6. Update the betting strategy for the next round and return to Step 1.

In the following sections, we describe in detail each component of the proposed adaptation scheme. Before proceeding, however, we pause to highlight the advantages of the matching loss (2) over entropy minimization.

3.2 Motivating example: entropy minimization vs. entropy matching

To facilitate the exposition of the proposed loss, it is useful to consider a toy, binary classification example with a one-dimensional input X in which we have oracle access both to the source P_{XY}^s and a fixed target distribution P_{XY}^t that does not vary over time. We commence by generating training data from P_{XY}^s, where $P(Y = 1) = P(Y = -1)$ and $P_{X|Y}^s = \mathcal{N}(Y^s, 1)$. See Figure 1 for an illustration of the source distribution. Throughout this experiment, we set the pre-trained Gaussian classifier $f_{\hat{\theta}}$ to be the Bayes optimal one for which $\hat{\theta} = 0$, and during test-time we optimize the parameter ω of the updated classifier. Since $\hat{\theta} = 0$, in this case $f_{\hat{\theta}+\omega}$ is simplified to f_ω. Further implementation details are provided in Appendix F.

As mentioned before, one of the advantages of our approach is its "no-harm" effect, i.e., when $P_{XY}^s = P_{XY}^t$ we ideally want to keep the decision boundary of the classifier intact. The red curve

[2] In the experiments in Section 4, we use a variation of this loss, described in Section 3.5.

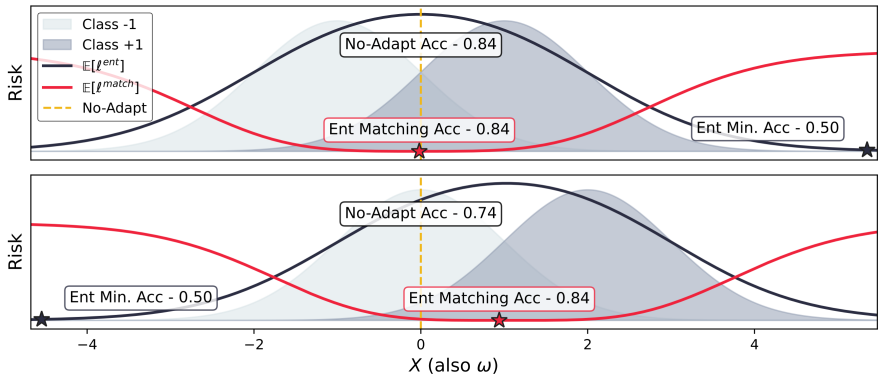

Figure 1: **Demonstration of the advantage of entropy matching on toy binary classification problem with Gaussian data.** The top panel represents an in-distribution setup in which $P^t_{XY} = P^s_{XY}$. The bottom panel illustrates an out-of-distribution setup, obtained by shifting the two Gaussians. The entropy matching (red) and entropy minimization (black) risks are presented as a function of ω. The dashed yellow line presents the decision boundary of the pre-trained classifier. The points marked by stars correspond to the decision boundary of the adapted classifiers.

in Figure 1 illustrates the entropy matching risk $\mathbb{E}_{X^t}[\ell^{\text{match}}(Z, \tilde{Z})]$ as a function of the classifier parameter ω. In this synthetic case, the ideal entropy matching risk can be evaluated since we have access to the generating distribution: we can obtain the ideal pseudo-entropy $\tilde{Z}^t$, given by $\tilde{Z} = F_s^{-1}(F_t(Z^t; f_\omega))$, where F_s and F_t are the CDF functions of the source and target entropy values, respectively; the formula above is nothing but the optimal transport map. Of course, in the practical online setting we consider in this work, F_t is unknown and varies over time. In fact, this is also true in the case studied here as the distribution of $Z^t = f_\omega(X^t)$ varies with ω, highlighting the importance of our online, adaptive testing procedure. Indeed, the ω obtained by our online adaptation scheme (`POEM`) minimizes the entropy matching risk and remains close to 0, as desired.

Meanwhile, the black curve in Figure 1 illustrates the values of the entropy risk $\mathbb{E}_{X^t}[\ell^{\text{ent}}(Z^t)]$ for varying ω. In contrast with our approach, the ω that minimizes the entropy risk is far from the optimal classifier. This approach results in a collapse towards a trivial classifier that always predicts -1, regardless of the value of X^t. Moving to an out-of-distribution scenario, where we consider test data sampled from a shifted version of the two Gaussians such that $Y^t = Y^s$ and $X^t = X^s + 1$. Following the bottom panel in Figure 1, it is evident that by minimizing the proposed entropy matching risk, the accuracy of the original (not adapted) classifier is effectively restored. In contrast, the entropy minimization paradigm once again collapses the model to make trivial predictions.

3.3 Online drift detection

We now turn to introduce a rigorous monitoring tool that is capable of detecting whether the distribution of the adapted classifier's entropy values Z^t, obtained at test time, deviate from Z^s—the source entropies obtained by applying the original model to the source data. To detect such shifts, we assume that we have access to F_s, the CDF of the source entropy values $Z^s = \ell^{\text{ent}}(f_{\hat{\theta}}(X^s))$. In practice, we estimate this CDF using holdout unlabeled samples X^s from the source distribution. Armed with F_s, we then apply the probability integral transform [96], allowing us to convert any sequence of i.i.d. entropy values from the source distribution $Z^s_1, Z^s_2, \ldots$ into a sequence of i.i.d. uniform random variables $F_s(Z^s_1), F_s(Z^s_2), \ldots$ on the $[0, 1]$ interval. Therefore, if we observe a sequence $F_s(Z^t_1), F_s(Z^t_2), \ldots$ of i.i.d. uniform variables at test time, we can infer that the target entropy distribution matches the source entropy distribution. Thus, if the sequence of transformed variables $F_s(Z^t_1), F_s(Z^t_2), \ldots$ deviates from the uniform distribution, we can infer that the corresponding target domain samples $Z^t_1, Z^t_2, \ldots$ differ from the source distribution. This observation lies at the core of our monitoring tool [61].

Specifically, we leverage the testing-by-betting approach to design a sequential test for the following null hypothesis:

$$\mathcal{H}_0 : u_j \triangleq F_s(Z^t_j) \sim \mathcal{U}[0, 1], \quad \forall j \in \mathbb{N}, \tag{3}$$

where $Z_j^t = \ell^{\text{ent}}(f_{\hat{\theta}+\omega_j}(X_j^t))$. In words, we continuously monitor the sequence of random variables $u_1, u_2, \ldots$ and test whether these deviate from the uniform null. We do so by formulating a test martingale, defined as follows.

Definition 1 (Test Martingale)**.** *A random process $\{S_j : j \in \mathbb{N}, S_0 = 1\}$ is a test martingale for the null hypothesis $\mathcal{H}_0$ if it satisfies the following:*

1. $S_j \geq 0 \quad \forall j \in \mathbb{N}$.
2. $\{S_j : j \in \mathbb{N}_0\}$ *is a martingale under $\mathcal{H}_0$, i.e.,* $\mathbb{E}_{\mathcal{H}_0}[S_j \mid S_1, \ldots, S_{j-1}] = S_{j-1}$.

The martingale can be thought of as the wealth process in the game-theoretical interpretation of the test, obtained by betting toy money against $\mathcal{H}_0$ as new data points arrive. We initialize this game with $S_0 = 1$, and update the wealth process as follows [61]:

$$S_j(u_j) \triangleq S_{j-1} \cdot b(u_j) \quad \text{where} \quad b(u_j) = 1 + \epsilon_j(u_j - 0.5). \tag{4}$$

Above, $b(u) \in [0, 2]$ is the *betting function*. The *betting variable* $\epsilon_j \in [-2, 2]$ controls how aggressive the bet is, and it can be determined based on past observations $u_1, \ldots, u_{j-1}$, as we detail later in this section. However, before introducing our strategy to update ϵ_j over time, we should first discuss the properties of the betting function $b(u_j)$. The idea behind this choice is that we sequentially test whether the sequence of $u_1, \ldots, u_j$ observed up to time step j has mean 0.5. Indeed, if the null is true, $\mathbb{E}_{\mathcal{H}_0}[u] = 0.5$ and thus $\mathbb{E}_{\mathcal{H}_0}[b(u)] = 1$. As a result, under the null, the martingale is unlikely to grow significantly beyond its initial value—this is a consequence of Ville's inequality; see Appendix D. However, if the null is false, we can gather evidence against the uniform null hypothesis by placing more aggressive bets, especially when past observed values $u_1, \ldots, u_{j-1}$ deviate significantly from 0.5. This highlights the role of ϵ_j, controlling the value and direction of our wagers in each round, betting on whether u_j would be over/under 0.5. Following (4), when ϵ_j and $(u_j - 0.5)$ have the same sign, we win the bet and obtain $b(u_j) > 1$. This implies that our capital increase as $S_j(u_j) := S_{j-1} \cdot b(u_j)$. Notice that, in this case, a larger ϵ_j (in absolute value) will allow us to increase the capital more rapidly, resulting in a more powerful test. However, if ϵ_j and $(u_j - 0.5)$ have different signs, we lose the bet and obtain $b(u_j) < 1$. Here, a larger ϵ_j would incur a more significant loss of capital. The challenge in choosing ϵ_j lies in the restriction that ϵ_j can only be determined based on past experience, i.e., we must set its value without looking at the new u_j. This restriction is crucial to ensure the validity of the martingale, as detailed in the following proposition.

Proposition 1. *The random process presented in* (4) *is a valid test martingale for $\mathcal{H}_0$* (3)*.*

The proof is given in Appendix C.1; it is a well-known result, see, e.g., [61]. This property is crucial to form the proposed online distributional matching mechanism, introduced in the next section. Appendix D provides further details on how the test martingale is used for distribution shift detection.

We now turn to present an adaptive approach to set the betting variable ϵ_j in a manner that enables powerful detection of drifting target entropies. This is especially important given the dynamic nature of both the target data and the continuous, online updates of the model. To achieve this, we adopt an online learning technique to learn ϵ_j from past observations, with the goal of maximizing the wealth by minimizing the negative log of the wealth process up to step j [70]:

$$-\log(S_j(u_j)) = -\log \prod_{\tau=1}^{j} b_\tau(u_\tau) = -\sum_{\tau=1}^{j} \log(b_\tau(u_\tau)) = -\sum_{\tau=1}^{j} \log(1 + \epsilon_\tau(u_\tau - 0.5)). \tag{5}$$

This formulation allows us to learn how to predict ϵ_j using past samples via gradient descent [70].

Specifically, our optimization approach relies on the scale-free online gradient descent (SF-OGD) algorithm [59]. Importantly, extending SF-OGD to our setting is not straightforward, since ϵ_j must be in the range of $[-2, 2]$ to form a valid test martingale. In the interest of space, we present this algorithm and its theoretical analysis in Appendix B and only highlight here its key feature. SF-OGD allows us to attain an anytime regret guarantee, which is presented formally in Theorem 1 of the Appendix. This guarantee bounds the difference between the negative log of the wealth process (i) obtained by the predicted ϵ_t over time horizon $1 \leq t \leq j$, and (ii) obtained by the best betting variable $\epsilon^\star$ that can only be calculated in hindsight, after looking at the data up to time $1 \leq t \leq j$. Informally, our theory shows this regret is bounded by $c \cdot \sqrt{t}$ for all $1 \leq t \leq j$, where the constant c depends on the problem parameters; it is formulated precisely in Theorem 1. In turn, the anytime regret guarantee confirms that our SF-OGD approach effectively learns ϵ_j, capturing the dynamic changes of both the target distribution and the model $f_{\hat{\theta}+\omega_j}$ in a fully online setting.

3.4 Online model adaptation

Having established a powerful betting strategy, we now turn to show how to transform the test martingale $\{S_j : j \in \mathbb{N}\}$ into a sequence of adapted pseudo-entropy values $\tilde{Z}_1, \tilde{Z}_2, ...$ that better match the distribution of Z^s. In what follows, we describe an algorithm to obtain $\tilde{Z}_j$, which draws inspiration from [61], and then connect this procedure to optimal transport.

Our adaptation scheme leverages the fact that any valid betting function is essentially a likelihood ratio [61, 82]. This property implies that our betting function $b(u_j) = 1 + \epsilon_j(u_j - 0.5)$ satisfies

$$b(u_j) = \frac{dQ(u_j)}{dG(u_j)}, \tag{6}$$

where $dQ(u_j)$ and $dG(u_j)$ are the densities of *some* alternative distribution Q and the null distribution G, respectively. In our case, the null distribution G is the uniform distribution Uniform$(0, 1)$, and the alternative distribution Q can be intuitively thought of as an approximation of the unknown target entropy's CDF; we formalize this intuition hereafter. Leveraging this likelihood ratio interpretation, we follow [61] and extract the alternative distribution Q by re-writing (6) as $dQ(u_j) = b(u_j) \cdot dG(u_j)$ and computing the integral:

$$Q(u_j) = \int_0^{u_j} b(v)dG(v)dv = \int_0^{u_j} b(v) \cdot 1 \cdot dv = \frac{1}{2}\epsilon_j \cdot u_j^2 + (1 - \frac{\epsilon_j}{2}) \cdot u_j. \tag{7}$$

Above, we used the fact that the null density is $dG(v)$ equals 1 over the support $[0, 1]$. Having access to Q, we can compute the adapted $\tilde{u}_j := Q(u_j)$ that can be intuitively interpreted as the result of applying the probability integral transform to Z_j^t using the estimated target entropy CDF. With this intuition in place, we can further convert $\tilde{u}_j$ into a pseudo-entropy value $\tilde{Z}_j^t$ that better matches the distribution of the source entropies. This is obtained by applying the inverse source CDF to $\tilde{u}_j$, resulting in $\tilde{Z}_j = F_s^{-1}(Q(u_j))$. Observe that we assume here that F_s is invertible, however, in practice, we compute the pseudo-inverse of F_s.

To connect the adaptation scheme presented above to the optimal transport map between the target and source entropies, it is useful to consider an ideal case where we use the log-optimal bet for testing a point null [82]. In our case, the null hypothesis is that the distribution of the source entropies Z^s and target entropies Z_j^t is the same, which implies $\mathcal{H}_0$ in (3). Following [82], the optimal bet for our null is the true likelihood ratio, formulated as

$$b_Z^{\text{opt}}(Z_j^t) \triangleq \frac{dF_t^j(Z_j^t)}{dF_s(Z_j^t)}, \tag{8}$$

where F_t^j is the CDF of the target entropy Z_j^t. To align with (6), we can equivalently write $b_Z^{\text{opt}}(Z_j^t)$ as a betting function that gets u_j as an input [61, Lemma 1]:

$$b_Z^{\text{opt}}(Z_j^t) = b_Z^{\text{opt}}(F_s^{-1}(u_j)) = \frac{dF_t^j(F_s^{-1}(u_j))}{dF_s(F_s^{-1}(u_j))} \triangleq \frac{dQ^{\text{opt}}(u_j)}{dG^{\text{opt}}(u_j)} = b_u^{\text{opt}}(u_j). \tag{9}$$

Notably, this optimal betting function is infeasible to compute in practice, as F_t^j is unknown. Yet, it implicitly suggests that more powerful betting functions could result in a better estimate of the target entropy CDF. Also, the optimal betting function reveals an important property of our adaptation scheme, formally given below.

Proposition 2. *Let X_j^t be a fresh sample from the target domain with its corresponding $Z_j^t = \ell^{\text{ent}}(f_{\hat{\theta}+\omega}(X_j^t))$ and $u_j = F_s(Z_j^t)$. Assume F_s is invertible and Z_j^t is continuous, and suppose the betting function represents the true likelihood ratio* (9)*. Then, the adapted $\tilde{Z}_j^t = F_s^{-1}(Q^{\text{opt}}(u_j))$ is the optimal transport map from target to source entropies with respect to the Wasserstein distance.*

The proof of this proposition builds on [61, Lemma 1] and is provided in Appendix C.4. This result highlights that our online, martingale-based adaptation scheme is grounded in optimal transport principles. This, in turn, provides a principled way to minimize the discrepancy between probability distributions. Leveraging this connection, the entropy matching loss function (2), which we employ to self-train the model, can be understood as minimizing the discrepancy between the entropy

distributions of the source and target domains. This implies that our loss function aligns the model's predictions across these domains. This connection also explains the "no-harm" effect of the proposed loss. When $P^s_{XY} = P^t_{XY}$ we get that $Q^{\text{opt}}(u_j) = G^{\text{opt}}(u_j) = u_j$ in the ideal case of (9), implying that $\tilde{Z}^t_j = Z^t_j$. In practice, considering the betting function from (4), we anticipate that ϵ_j will be close to zero thanks to our online optimization scheme, which, in turn, results in $Q(u_j) \approx u_j$ in (7).

3.5 Putting it all together

Algorithm 2 in the Appendix summarizes the entire adaptation process of `POEM`. This algorithm starts by computing the empirical CDF $\hat{F}_s$ of the source entropies to estimate F_s, using unlabeled holdout samples from the source domain (line 6). The betting and pseudo-entropy estimation steps are presented in lines 12–14. Observe that in line 14 we use the pseudo-inverse of $\hat{F}_s$ to obtain $\tilde{Z}_j$. The algorithm then proceeds to adapt the classifier's parameters in a direction that minimizes our self-supervised loss (line 15). Specifically, we only update the parameters of the normalization layers ω of the classifier $f_{\hat{\theta}+\omega}$, which is a common practice in the test-time adaptation literature [41–43]. The self-training step is done by minimizing a variation of the entropy matching loss ℓ^{match} (2):

$$\ell^{\text{match++}}_{\lambda}(Z^t_j, \tilde{Z}_j) = \ell^{\text{match}}(Z^t_j, \tilde{Z}_j) \cdot \frac{\mathbb{1}[Z^t_j < \lambda]}{\exp\{2 \cdot (Z^t_j - \lambda)\}}, \text{ where } Z^t_j = \ell^{\text{ent}}(f_{\hat{\theta}+\omega_j}(X^t_j)). \quad (10)$$

The above loss function includes an additional sample-filtering $\mathbb{1}[Z^t_j < \lambda]$ and sample-weighting $1/\exp\{2 \cdot (Z^t_j - \lambda)\}$ components, where $\lambda > 0$ is a pre-defined thresholding parameter. The sample-filtering idea is widely used in this literature [42, 43], as the predictions of samples with high entropy examples tend to be inaccurate. Aligning with this intuition, the sample-weighting gives a higher weight to samples with low entropies [42]. Finally, we predict a new betting parameter ϵ_j for the next iteration by applying an SF-OGD step (line 16).

4 Experiments

We conduct a comprehensive evaluation of POEM across a diverse range of datasets and scenarios commonly used in test-time adaptation literature. Our experiments span ImageNet, ImageNet-C, CIFAR10-C, and CIFAR100-C datasets for evaluating the robustness to shifts induced by corruptions, and the Office-Home dataset for domain adaptation. We study the performance of our method in both single-shift and continual-shift settings. Our evaluation demonstrates that `POEM` is highly competitive with leading baseline test-time adaptation methods in terms of accuracy and runtime. In the interest of space, this section focuses on the results for the ImageNet dataset, as it is the most challenging one among those considered. Details and results for the experiments on CIFAR and Office-Home datasets are provided in Appendices F.3 and F.4, respectively.

Throughout this section we use the test set of ImageNet to form our in-distribution dataset, and utilize ImageNet-C—which contains 15 different types of corruptions at five increasing severity levels—to simulate various out-of-distribution scenarios. Notably, since the images in ImageNet-C are variations of the same images from the ImageNet test set, our experiments simulate a realistic out-of-distribution test set by including only a single corrupted version of each image. To demonstrate the versatility of `POEM`, we consider two pre-trained classifiers $f_{\hat{\theta}}$ of different architectures: Vision Transformer (ViT) [63] with layer norm (LN) and ResNet50 [64] with group norm (GN). We compare `POEM` to four leading entropy minimization methods—`TENT` [41], `EATA` [42], `SAR` [43], and `COTTA` [97]—using code provided by the authors. Importantly, all adaptation methods update only the normalization parameters (LN/GN) of the model, ensuring a fair comparison. Following [43], we employ a fully online setting with a batch size of 1, in which the model is updated after observing a new test sample; see Appendix F.2 for implementation details and choice of hyper-parameters. In the interest of space, we defer the results obtained by the ResNet classifier to Appendix F.2.1 and focus here on the results obtained by the ViT model.

Continual shifts Inspired by [97, 98], we evaluate our approach in a continual setup in which sudden distribution shifts occur during testing. To simulate this, we create a test set of 15,000 samples by randomly selecting 1,000 samples from each corruption type at a fixed severity level (a corruption segment) and concatenating all 15 segments to form the test data. We apply all adaptation methods

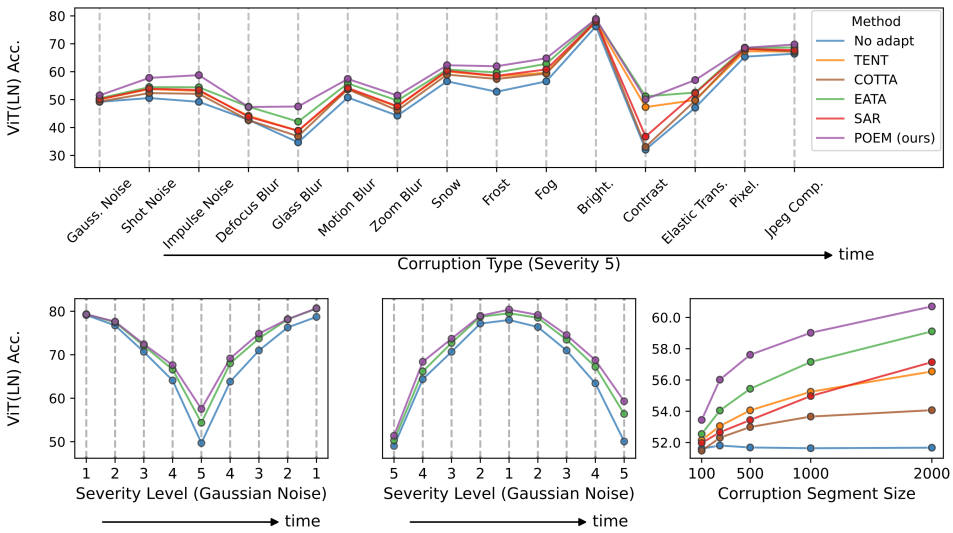

Figure 2: **Continual test-time adaptation on ImageNet-C with a ViT model**. **Top:** Per-corruption accuracy with a corruption segment size of 1,000 examples. Results are obtained over 10 independent trials; error bars are tiny. **Bottom left and center:** Severity shift—low (1) to high (5) and back to low (left), and high (5) to low (1) and back to high (center). To improve the readability here, we only present `POEM`, the best-performing baseline method (`EATA`), and the `no-adapt` approach. **Bottom right:** Mean accuracy under continual corruptions as a function of the corruption segment size.

in combination with a ViT model and present the results in Figure 2. Following the bottom panel in that figure, one can see that `POEM` achieves higher accuracy than all of the benchmark methods. Next, we investigate how quickly our method adapts the model to new shifts by varying the segment size of each corruption. As shown in Figure 2 (bottom right), `POEM` exhibits faster adaptation than the baseline methods. It successfully enhances the accuracy of the pre-trained model with as few as 100 examples per corruption (test set of size 1,500) as well as with longer adaptation using 2,000 examples per corruption (test set of size 30,000). Finally, we explore another realistic scenario of continual adaptation by varying the severity level every 1,000 samples while keeping a fixed corruption type. The bottom left and center panels in Figure 2 show that `POEM` outperforms the best baseline method (`EATA`) in this setting as well. Lastly, similar experiments conducted with a ResNet model are presented in Figure 5 in the Appendix, showing that our method attains faster adaptation and superior accuracy than the baseline methods.

Single shift We now consider a scenario with a single corruption type of a fixed severity level, which follows [41, 43, 99]. Table 1 summarizes the average results across all corruptions for severity level 5, demonstrating that `POEM` achieves an average accuracy of 67.36% for the ViT model, outperforming the best baseline method (`EATA`) with an absolute average accuracy gap of 3.22%. A detailed breakdown by corruption type for each classifier is provided in Table 2 in the Appendix. Notably, `POEM` outperforms all benchmark methods on all corruption types for ViT, while achieving higher test accuracy in 9 out of the 15 corruption types for the ResNet model.

In distribution In this setting, we apply all methods on the validation set of the ImageNet dataset. Following Table 3 in the Appendix, all the methods maintain a similar accuracy as the original model, however, the baseline methods tend to increase the expected calibration error (ECE) [68, 100] and unnecessarily modify the model's parameters, as measured by $\|\omega\|_2^F$. In contrast, following Figure 3 (left panel), `POEM` exhibits minimal changes both for ECE and model parameters, as desired. Figure 6 in the Appendix leads to similar conclusions for the ResNet model.

Additional experiments on ImageNet, including ablation study The right panel of Figure 3 plots the value of the betting parameter ϵ over time, for both in- and out-of-distribution scenarios. Observe how ϵ remains near zero under the in-distribution setting, explaining the minimal change in accuracy, ECE, and model's parameters $\|\omega\|_F^2$, presented in the left panel of Figure 3. By contrast, when

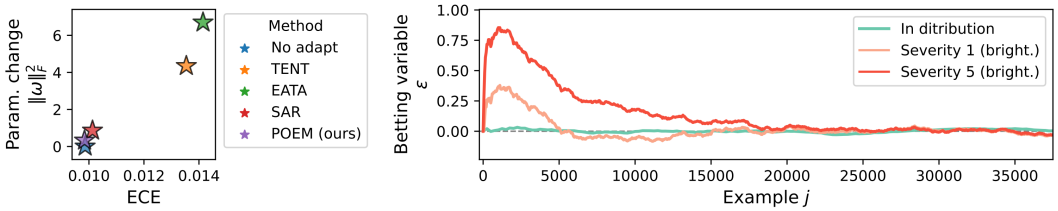

Figure 3: **In-distribution experiment on ImageNet (left panel)**: calibration error (ECE [100]) versus $\|\omega\|_F^2$—a metric that evaluates the classifier's parameters deviation from the original ViT model. Lower values on both axes are better. Results are averaged across 10 independent trials; standard errors and accuracy of each method are reported in Table 3 in the appendix. **The behavior of the betting parameter (right panel)**: the value of ϵ is presented as a function of time for both in- and out-of-distribution experiments (a single shift, two severity levels).

considering out-of-distribution test data of a single shift, we can see that ϵ is high at the beginning and gradually reduces over time, indicating that the self-training process adapts the model to the new environment. A similar behavior is observed for the ResNet model; see Figure 6 in the Appendix. This conclusion is further supported by Figure 7 in the Appendix, showing that the CDF of the target entropies of the adapted ResNet model nearly matches the CDF of the source entropies. In Figure 4 of the Appendix we show how the martingale can powerfully detect shifts—even a minor one (brightness with severity level 1)—and also show how our adaptation mechanism gradually limits the martingale's growth. Lastly, we conduct an ablation study, comparing the test accuracy of `POEM` using two loss functions: ℓ^{match} (2) and $\ell^{\text{match++}}$ (10). Table 4 in the Appendix presents these results for a single shift scenario, averaged over 15 corruptions of ImageNet-C at the highest severity level. Both losses improve the original model accuracy, with $\ell^{\text{match++}}$ showing better adaptation performance.

Experiments on CIFAR-10C, CIFAR100-C, and OfficeHome datasets, sensitivity study, and runtime comparisons Appendices F.3 and F.4 present experiments on these additional datasets, leading to similar conclusions about the competitive adaptation accuracy of `POEM` compared to baseline methods. Notably, Appendix F.3 includes experiments on both relatively short and long adaptation streams, with lengths of 15,000 and 112,500 samples, respectively. These experiments also include a sensitivity study on the learning rate η used for self-training the model, showing that `POEM` exhibits greater stability across different learning rate values compared to `SAR` and `TENT`. Additionally, these experiments show that `POEM`'s runtime is comparable to `TENT` and `EATA`, and faster than `SAR`.

5 Discussion

We introduced a novel, martingale-based approach for test-time adaptation that drives the test-time entropies of the self-trained model to match the distribution of source entropies. We validated our approach with numerical experiments, demonstrating that: (i) under in-distribution settings, `POEM` maintains the performance of the source model while avoiding overconfident predictions, a crucial advantage over entropy minimization methods; (ii) in relatively short out-of distribution test periods, our approach achieves faster adaptation than entropy minimization methods, which is attributed to our betting scheme that quickly reacts to distribution shifts; and (iii) in extended test periods, `POEM` achieves comparable adaptation performance and stability to strong baseline methods.

One limitation of our method is the requirement for holdout unlabeled source domain data to estimate the source CDF. Notably, this CDF is pre-computed offline, and at test time we do not require additional access to source data, similar to `EATA`'s requirements. Another limitation of our approach is the choice of hyperparameters, particularly the self-training learning rate η. However, our sensitivity study showed that our method is fairly robust to this choice, especially compared to baseline methods. Lastly, similar to other experimental works, we anticipate that `POEM` may fail to improve accuracy in settings that we have not explored, especially when facing an aggressive shift. Yet, our monitoring tool can detect such distribution shifts, which is an important mechanism that does not appear in other test-time adaptation methods.

Future directions are discussed in Section G of the Appendix. Lastly, we note that while the goal of this paper is to enhance the robustness of ML to unseen environments, there are many potential societal consequences of our method, similar to other works that aim to advance this field.

6 Acknowledgments and Disclosure of Funding

This research was supported by the ISRAEL SCIENCE FOUNDATION (grant No. 729/21). Y. R. thanks the Career Advancement Fellowship, Technion.

References

[1] Qin Wang, Olga Fink, Luc Van Gool, and Dengxin Dai. Continual test-time domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7201–7211, June 2022.

[2] Thomas Varsavsky, Mauricio Orbes-Arteaga, Carole H Sudre, Mark S Graham, Parashkev Nachev, and M Jorge Cardoso. Test-time unsupervised domain adaptation. In *International Conference in Medical Image Computing and Computer Assisted Intervention*, pages 428–436. Springer, 2020.

[3] Abolfazl Farahani, Sahar Voghoei, Khaled Rasheed, and Hamid R Arabnia. A brief review of domain adaptation. *Advances in data science and information engineering*, pages 877–894, 2021.

[4] Garrett Wilson and Diane J Cook. A survey of unsupervised deep domain adaptation. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 11(5):pp. 1–46, 2020.

[5] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *Proceedings of the International Conference on Machine Learning*, pages 1180–1189. PMLR, 2015.

[6] Vishal M Patel, Raghuraman Gopalan, Ruonan Li, and Rama Chellappa. Visual domain adaptation: A survey of recent advances. *IEEE signal processing magazine*, 32(3):pp. 53–69, 2015.

[7] Wouter M Kouw and Marco Loog. An introduction to domain adaptation and transfer learning. *arXiv preprint arXiv:1812.11806*, 2018.

[8] Xingchao Peng, Ben Usman, Neela Kaushik, Dequan Wang, Judy Hoffman, and Kate Saenko. Visda: A synthetic-to-real benchmark for visual domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 2021–2026, 2018.

[9] Kun Zhang, Bernhard Schölkopf, Krikamol Muandet, and Zhikun Wang. Domain adaptation under target and conditional shift. In *Proceedings of the International Conference on Machine Learning*, pages 819–827. PMLR, 2013.

[10] Vidit Jain and Erik Learned-Miller. Online domain adaptation of a pre-trained cascade of classifiers. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 577–584. IEEE, 2011.

[11] Xiaofeng Liu, Chaehwa Yoo, Fangxu Xing, Hyejin Oh, Georges El Fakhri, Je-Won Kang, and Jonghye Woo. Deep unsupervised domain adaptation: A review of recent advances and perspectives. *APSIPA Transactions on Signal and Information Processing*, 11, 2022.

[12] John Blitzer, Ryan McDonald, and Fernando Pereira. Domain adaptation with structural correspondence learning. In *Proceedings Conference on Empirical Methods in Natural Language Processing*, pages 120–128, 2006.

[13] Yongjie Shi, Xianghua Ying, and Jinfa Yang. Deep unsupervised domain adaptation with time series sensor data: A survey. *Sensors*, 22(15), 2022.

[14] Joaquin Quionero-Candela, Masashi Sugiyama, Anton Schwaighofer, and Neil D. Lawrence. *Dataset Shift in Machine Learning*. The MIT Press, 2009.

[15] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *International Conference on Learning Representations*, 2018.

[16] Jeya Maria Jose Valanarasu, Pengfei Guo, Vibashan VS, and Vishal M. Patel. On-the-fly test-time adaptation for medical image segmentation. *arXiv preprint arXiv:2203.05574*, 2022.

[17] Yanyu Ye, Zhenxi Zhang, Wei Wei, and Chunna Tian. Multi task consistency guided source-free test-time domain adaptation medical image segmentation. *arXiv preprint arXiv:2310.11766*, 2023.

[18] Neerav Karani, Ertunc Erdil, Krishna Chaitanya, and Ender Konukoglu. Test-time adaptable neural networks for robust medical image segmentation. *Medical Image Analysis*, 68:101907, 2021.

[19] Yufan He, Aaron Carass, Lianrui Zuo, Blake E Dewey, and Jerry L Prince. Autoencoder based self-supervised test-time adaptation for medical image analysis. *Medical Image Analysis*, 72:102136, 2021.

[20] Zhang Li, Zheng Zhong, Yang Li, Tianyu Zhang, Liangxin Gao, Dakai Jin, Yue Sun, Xianghua Ye, Li Yu, Zheyu Hu, Jing Xiao, Lingyun Huang, and Yuling Tang. From community-acquired pneumonia to COVID-19: a deep learning–based method for quantitative analysis of COVID-19 on thick-section CT scans. *European Radiology*, 30(12):6828–6837, 2020.

[21] Jian Liang, Ran He, and Tieniu Tan. A comprehensive survey on test-time adaptation under distribution shifts. *arXiv preprint arXiv:2303.15361*, 2023.

[22] Prashant Pandey, Mrigank Raman, Sumanth Varambally, and Prathosh AP. Generalization on unseen domains via inference-time label-preserving target projections. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12924–12933, June 2021.

[23] Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, Dawn Song, Jacob Steinhardt, and Justin Gilmer. The many faces of robustness: A critical analysis of out-of-distribution generalization. *ICCV*, 2021.

[24] Marvin Zhang, Sergey Levine, and Chelsea Finn. MEMO: Test time robustness via adaptation and augmentation. *Advances in neural information processing systems*, 35:38629–38642, 2022.

[25] Xuefeng Hu, Gokhan Uzunbas, Sirius Chen, Rui Wang, Ashish Shah, Ram Nevatia, and Ser-Nam Lim. Mixnorm: Test-time adaptation through online normalization estimation. *arXiv preprint arXiv:2110.11478*, 2021.

[26] Saypraseuth Mounsaveng, Florent Chiaroni, Malik Boudiaf, Marco Pedersoli, and Ismail Ben Ayed. Bag of tricks for fully test-time adaptation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1936–1945, 2024.

[27] Eric Mintun, Alexander Kirillov, and Saining Xie. On interaction between augmentations and corruptions in natural corruption robustness. *Advances in Neural Information Processing Systems*, 34:3571–3583, 2021.

[28] Liang Chen, Yong Zhang, Yibing Song, Ying Shan, and Lingqiao Liu. Improved test-time adaptation for domain generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 24172–24182, 2023.

[29] Minmin Chen, Kilian Q Weinberger, and John Blitzer. Co-training for domain adaptation. *Advances in neural information processing systems*, 24:2456–2464, 2011.

[30] Yuanyuan Xu, Meina Kan, Shiguang Shan, and Xilin Chen. Mutual learning of joint and separate domain alignments for multi-source domain adaptation. In *WACV*, pages 1890–1899, 2022.

[31] Yang Zou, Zhiding Yu, BVK Kumar, and Jinsong Wang. Unsupervised domain adaptation for semantic segmentation via class-balanced self-training. In *Proceedings of the European conference on computer vision (ECCV)*, pages 289–305, 2018.

[32] Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *Proceedings of the International Conference on Machine Learning*, pages 9229–9248. PMLR, 2020.

[33] Jian Shen, Yanru Qu, Weinan Zhang, and Yong Yu. Wasserstein distance guided representation learning for domain adaptation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[34] Yuejiang Liu, Parth Kothari, Bastien van Delft, Baptiste Bellot-Gurlet, Taylor Mordan, and Alexandre Alahi. TTT++: When does self-supervised test-time training fail or thrive? *Advances in neural information processing systems*, 2021.

[35] Yves Grandvalet and Yoshua Bengio. Semi-supervised learning by entropy minimization. *Advances in neural information processing systems*, 17, 2004.

[36] Philip Haeusser, Thomas Frerix, Alexander Mordvintsev, and Daniel Cremers. Associative domain adaptation. In *Proceedings of the IEEE international conference on computer vision*, pages 2765–2773, 2017.

[37] Eric Tzeng, Judy Hoffman, Trevor Darrell, and Kate Saenko. Simultaneous deep transfer across domains and tasks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 4068–4076, 2015.

[38] Fabio Maria Carlucci, Lorenzo Porzi, Barbara Caputo, Elisa Ricci, and Samuel Rota Bulo. AutoDIAL: Automatic domain alignment layers. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 5077–5085, 2017.

[39] Kuniaki Saito, Yoshitaka Ushiku, and Tatsuya Harada. Asymmetric tri-training for unsupervised domain adaptation. In *Proceedings of the International Conference on Machine Learning*, pages 2988–2997, 2017.

[40] Rui Shu, Hung H Bui, Hirokazu Narui, and Stefano Ermon. A DIRT-T approach to unsupervised domain adaptation. In *Proceedings of the International Conference on Learning Representations*, 2018.

[41] Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. In *International Conference on Learning Representations*, 2020.

[42] Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Yaofo Chen, Shijian Zheng, Peilin Zhao, and Mingkui Tan. Efficient test-time model adaptation without forgetting. In *Proceedings of the International Conference on Machine Learning*, pages 16888–16905, 2022.

[43] Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Zhiquan Wen, Yaofo Chen, Peilin Zhao, and Mingkui Tan. Towards stable test-time adaptation in dynamic wild world. In *International Conference on Learning Representations*, 2023.

[44] Masud An-Nur Islam Fahim and Jani Boutellier. SS-TTA: Test-time adaption for self-supervised denoising methods. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1178–1187, 2023.

[45] Dian Chen, Dequan Wang, Trevor Darrell, and Sayna Ebrahimi. Contrastive test-time adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 295–305, 2022.

[46] Tomer Cohen, Noy Shulman, Hai Morgenstern, Roey Mechrez, and Erez Farhan. Self-supervised dynamic networks for covariate shift robustness. *arXiv preprint arXiv:2006.03952*, 2020.

[47] Alexander Bartler, Andre Bühler, Felix Wiewel, Mario Döbler, and Bin Yang. Mt3: Meta test-time training for self-supervised test-time adaption. *International Conference on Artificial Intelligence and Statistics*, pages 3080–3090, 2022.

[48] Alexander Bartler, Florian Bender, Felix Wiewel, and Bin Yang. TTAPS: Test-time adaption by aligning prototypes using self-supervision. In *IEEE International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2022.

[49] Hao Zhao, Yuejiang Liu, Alexandre Alahi, and Tao Lin. On pitfalls of test-time adaptation. In *International Conference on Machine Learning (ICML)*, 2023.

[50] Yi Su, Yixin Ji, Juntao Li, Hai Ye, and Min Zhang. Beware of model collapse! fast and stable test-time adaptation for robust question answering. In *Conference on Empirical Methods in Natural Language Processing*, 2023.

[51] Taesik Gong, Yewon Kim, Taeckyung Lee, Sorn Chottananurak, and Sung-Ju Lee. SoTTA: Robust test-time adaptation on noisy data streams. *Advances in Neural Information Processing Systems*, 36, 2024.

[52] Ori Press, Ravid Shwartz-Ziv, Yann LeCun, and Matthias Bethge. The entropy enigma: Success and failure of entropy minimization. *arXiv preprint arXiv:2405.05012*, 2024.

[53] Xiaofu Wu, Suofei Zhang, Quan Zhou, Zhen Yang, Chunming Zhao, and Longin Jan Latecki. Entropy minimization versus diversity maximization for domain adaptation. *IEEE Transactions on Neural Networks and Learning Systems*, 34(6):2896–2907, 2021.

[54] Pietro Morerio, Jacopo Cavazza, and Vittorio Murino. Minimal-entropy correlation alignment for unsupervised deep domain adaptation. In *International Conference on Learning Representations*, 2018.

[55] Glenn Shafer, Alexander Shen, Nikolai Vereshchagin, and Vladimir Vovk. Test martingales, bayes factors and p-values. *Statistical Science*, 26(1), February 2011.

[56] Vladimir Vovk, Ilia Nouretdinov, and Alexander Gammerman. Testing exchangeability on-line. In *Proceedings of the 20th International Conference on Machine Learning (ICML)*, pages 768–775, 2003.

[57] Vladimir Vovk, Ivan Petej, and Alex Gammerman. Protected probabilistic classification. In *Conformal and Probabilistic Prediction and Applications*, pages 297–299, 2021.

[58] Francesco Orabona and Dávid Pál. Scale-free algorithms for online linear optimization. In *International Conference on Algorithmic Learning Theory*, pages 287–301, 2015.

[59] Francesco Orabona and Dávid Pál. Scale-free online learning. *Theoretical Computer Science*, 716:50–69, 2018.

[60] Francesco Orabona and Dávid Pál. Coin betting and parameter-free online learning. *Advances in Neural Information Processing Systems*, 29:577–585, 2016.

[61] Vladimir Vovk. Protected probabilistic regression. Technical report, Tech. Rep, 2021.

[62] Cédric Villani. *Optimal transport: old and new*, volume 338. Springer, 2009.

[63] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2020.

[64] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[65] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *International Conference on Learning Representations*, 2018.

[66] Hemanth Venkateswara, Jose Eusebio, Shayok Chakraborty, and Sethuraman Panchanathan. Deep hashing network for unsupervised domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5018–5027, 2017.

[67] Nicolas Courty, Rémi Flamary, Devis Tuia, and Alain Rakotomamonjy. Optimal transport for domain adaptation. *IEEE transactions on pattern analysis and machine intelligence*, 39(9):1853–1865, 2016.

[68] Eungyeup Kim, Mingjie Sun, Aditi Raghunathan, and J Zico Kolter. Reliable test-time adaptation via agreement-on-the-line. In *NeurIPS Workshop on Distribution Shifts: New Frontiers with Foundation Models*, 2023.

[69] Glenn Shafer and Vladimir Vovk. Game-theoretic foundations for probability and finance. *Wiley Series in Probability and Statistics*, 2019.

[70] Shubhanshu Shekhar and Aaditya Ramdas. Nonparametric two-sample testing by betting. *IEEE Transactions on Information Theory*, 2023.

[71] Aleksandr Podkopaev and Aaditya Ramdas. Sequential predictive two-sample and independence testing. *Advances in Neural Information Processing Systems*, 36, 2024.

[72] Shalev Shaer, Gal Maman, and Yaniv Romano. Model-X sequential testing for conditional independence via testing by betting. In *International Conference on Artificial Intelligence and Statistics*, pages 2054–2086, 2023.

[73] Peter Grünwald, Alexander Henzi, and Tyron Lardy. Anytime-valid tests of conditional independence under model-X. *Journal of the American Statistical Association*, pages 1–12, 2023.

[74] Valentina Fedorova, Alex Gammerman, Ilia Nouretdinov, and Vladimir Vovk. Plug-in martingales for testing exchangeability on-line. In *Proceedings of the International Conference on Machine Learning*, pages 923–930, 2012.

[75] Vladimir Vovk, Alexander Gammerman, and Glenn Shafer. Testing exchangeability. In *Algorithmic Learning in a Random World*, pages 227–263. Springer, 2022.

[76] Boyan Duan, Aaditya Ramdas, and Larry Wasserman. Interactive rank testing by betting. In *Conference on Causal Learning and Reasoning*, pages 201–235, 2022.

[77] Aaditya Ramdas, Johannes Ruf, Martin Larsson, and Wouter M Koolen. Testing exchangeability: Fork-convexity, supermartingales and e-processes. *International Journal of Approximate Reasoning*, 141:83–109, 2022.

[78] Aytijhya Saha and Aaditya Ramdas. Testing exchangeability by pairwise betting. In *International Conference on Artificial Intelligence and Statistics*, pages 4915–4923, 2024.

[79] Peter Grünwald, Rianne de Heide, and Wouter M Koolen. Safe testing. In *IEEE Information Theory and Applications Workshop (ITA)*, pages 1–54, 2020.

[80] Glenn Shafer, Alexander Shen, Nikolai Vereshchagin, and Vladimir Vovk. Test martingales, bayes factors and p-values. *Statistical Science*, 26(1):84, 2011.

[81] Vladimir Vovk and Ruodu Wang. E-values: Calibration, combination and applications. *The Annals of Statistics*, 49(3):1736–1754, 2021.

[82] Aaditya Ramdas, Peter Grünwald, Vladimir Vovk, and Glenn Shafer. Game-theoretic statistics and safe anytime-valid inference. *Statistical Science*, 38(4):576–601, 2023.

[83] Steven R Howard, Aaditya Ramdas, Jon McAuliffe, and Jasjeet Sekhon. Time-uniform, nonparametric, nonasymptotic confidence sequences. *The Annals of Statistics*, 49(2), 2021.

[84] Ian Waudby-Smith and Aaditya Ramdas. Estimating means of bounded random variables by betting. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 2023.

[85] Peter D Grünwald. The e-posterior. *Philosophical Transactions of the Royal Society A*, 381(2247), 2023.

[86] Muriel Felipe Pérez-Ortiz, Tyron Lardy, Rianne de Heide, and Peter Grünwald. E-statistics, group invariance and anytime valid testing. *arXiv preprint arXiv:2208.07610*, 2022.

[87] Wouter M Koolen and Peter Grünwald. Log-optimal anytime-valid e-values. *International Journal of Approximate Reasoning*, 141:69–82, 2022.

[88] Vladimir Vovk and Ruodu Wang. Confidence and discoveries with e-values. *Statistical Science*, 38(2):329–354, 2023.

[89] Shubhanshu Shekhar and Aaditya Ramdas. Reducing sequential change detection to sequential estimation. *arXiv preprint arXiv:2309.09111*, 2023.

[90] Jaehyeok Shin, Aaditya Ramdas, and Alessandro Rinaldo. E-detectors: A nonparametric framework for sequential change detection. *The New England Journal of Statistics in Data Science*, pages 1–32, 2023.

[91] Vladimir Vovk. Testing randomness online. *Statistical Science*, 36(4):595–611, 2021.

[92] Charalambos Eliades and Harris Papadopoulos. A conformal martingales ensemble approach for addressing concept drift. In *Conformal and Probabilistic Prediction with Applications*, volume 204, pages 328–346. PMLR, 2023.

[93] Vladimir Vovk. Testing for concept shift online. *arXiv preprint arXiv:2012.14246*, 2020.

[94] Liang Dai and Mohamed-Rafik Bouguelia. Testing exchangeability with martingale for change-point detection. *International Journal of Ambient Computing and Intelligence (IJACI)*, 12(2):1–20, 2021.

[95] Vladimir Vovk, Ivan Petej, Ilia Nouretdinov, Ernst Ahlberg, Lars Carlsson, and Alex Gammerman. Retrain or not retrain: Conformal test martingales for change-point detection. In *Conformal and Probabilistic Prediction and Applications*, pages 191–210. PMLR, 2021.

[96] Paul Lévy. Théorie de l'addition de variables aléatoires. second edition 1954. (gauthier-villars, paris). *The Mathematical Gazette*, 39, 1955.

[97] Qin Wang, Olga Fink, Luc Van Gool, and Dengxin Dai. Continual test-time domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7201–7211, 2022.

[98] Longhui Yuan, Binhui Xie, and Shuang Li. Robust test-time adaptation in dynamic scenarios. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15922–15932, 2023.

[99] Zixin Wang, Yadan Luo, Liang Zheng, Zhuoxiao Chen, Sen Wang, and Zi Huang. In search of lost online test-time adaptation: A survey. *arXiv preprint arXiv:2310.20199*, 2023.

[100] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International Conference on Machine Learning*, pages 1321–1330. PMLR, 2017.

[101] Menglong Lu, Zhen Huang, Zhiliang Tian, Yunxiang Zhao, Xuanyu Fei, and Dongsheng Li. Meta-tsallis-entropy minimization: a new self-training approach for domain adaptation on text classification. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 5159–5169, 2023.

[102] Hong Liu, Jianmin Wang, and Mingsheng Long. Cycle self-training for domain adaptation. *Advances in Neural Information Processing Systems*, 34:22968–22981, 2021.

[103] Dong-Hyun Lee et al. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, volume 3, page 896, 2013.

[104] Sachin Goyal, Mingjie Sun, Aditi Raghunathan, and J Zico Kolter. Test time adaptation via conjugate pseudo-labels. *Advances in Neural Information Processing Systems*, 35:6204–6218, 2022.

[105] Jun-Kun Wang and Andre Wibisono. Towards understanding GD with hard and conjugate pseudo-labels for test-time adaptation. In *International Conference on Learning Representations*, 2022.

[106] Steffen Schneider, Evgenia Rusak, Luisa Eck, Oliver Bringmann, Wieland Brendel, and Matthias Bethge. Improving robustness against common corruptions by covariate shift adaptation. *Advances in neural information processing systems*, 33:11539–11551, 2020.

[107] Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *International conference on machine learning*, pages 9229–9248. PMLR, 2020.

[108] Xinyang Chen, Sinan Wang, Jianmin Wang, and Mingsheng Long. Representation subspace distance for domain adaptation regression. In *ICML*, pages 1749–1759, 2021.

[109] Karsten M Borgwardt, Arthur Gretton, Malte J Rasch, Hans-Peter Kriegel, Bernhard Schölkopf, and Alex J Smola. Integrating structured biological data by kernel maximum mean discrepancy. *Bioinformatics*, 22(14):e49–e57, 2006.

[110] Aadyot Bhatnagar, Huan Wang, Caiming Xiong, and Yu Bai. Improved online conformal prediction via strongly adaptive online learning. In *Proceedings of the International Conference on Machine Learning*, pages 2337–2363. PMLR, 2023.

[111] Victor M Panaretos and Yoav Zemel. Statistical aspects of Wasserstein distances. *Annual review of statistics and its application*, 6:405–431, 2019.

[112] Jean Ville. *1ere these: Etude critique de la notion de collectif; 2eme these: La transformation de Laplace*. PhD thesis, Gauthier-Villars & Cie, 1939.

[113] Taesik Gong, Jongheon Jeong, Taewon Kim, Yewon Kim, Jinwoo Shin, and Sung-Ju Lee. Note: Robust continual test-time adaptation against temporal correlation. *Advances in Neural Information Processing Systems*, 35:27253–27266, 2022.

[114] Aleksandr Podkopaev and Aaditya Ramdas. Distribution-free uncertainty quantification for classification under label shift. In *Uncertainty in artificial intelligence*, pages 844–853. PMLR, 2021.

[115] Zhi Zhou, Lan-Zhe Guo, Lin-Han Jia, Dingchu Zhang, and Yu-Feng Li. Ods: Test-time adaptation in the presence of open-world data shift. In *International Conference on Machine Learning*, pages 42574–42588. PMLR, 2023.

A Additional related work: test-time adaptation

As discussed in the main manuscript, entropy minimization is known to be effective for test time adaptation. However, the works in [42, 43] demonstrate that samples with high entropy loss can lead to noisy or overly aggressive updates of model parameters. To address this issue, [42, 43] filter out high-entropy samples, and [43] also employs gradient clipping to stabilize self-training. These algorithmic modifications emphasize the importance of controlling the minimization of the entropy loss, which greatly aligns with the goal of our work. In our framework, we also use entropy as a guiding force for self-training. However, instead of directly minimizing the entropy, our approach focuses on matching the distribution of the entropy losses at test time with that of the source domain. Notably, our method is versatile and can accommodate alternative choices beyond entropy, such as Tsallis entropy [101, 102] or cross-entropy evaluated with a pseudo-label [103–105] in place of the unknown true label. We leave the exploration of these alternative options for future work.

Another concern in test-time adaptation frameworks is the continuous learning mechanism, which often leads to performance degradation on in-distribution data. To address this challenge, `EATA` [42] introduces an anti-forgetting strategy that optimizes the model by focusing on the reliability of samples and incorporates a weight regularizer to further improve stability. In our work, instead of relying on weight regularization, we preserve in-distribution performance through a "no-harm" approach, ensuring minimal model updates where no distribution shifts have occurred.

A crucial aspect of self-training is the selection of model parameters ω to update. A widely adopted practice is to update the parameters of the normalization layers. This constraint plays a vital role in mitigating overfitting, which is imperative to prevent the model from collapsing and making trivial predictions. The `TENT` [41] approach demonstrates that minimizing entropy by modifying only the batch normalization parameters can significantly enhance out-of-distribution performance. However, `TENT` requires working with large batches, which can limit the ability to handle mixed-distribution shifts within a single batch—for instance, a batch containing a subset of blurred images, another subset of noisy images, and so on. To alleviate this issue, the `SAR` method suggests updating the group or layer normalization parameters, unlocking the use of smaller batch sizes. In turn, this adjustment enhances the model's adaptivity under mixed-distribution shifts. In our work, we follow this line of research and update the layer normalization parameters; however, we optimize a completely different loss function, aiming to match the distributions of the source and target entropies. While there are works that suggest matching between the source and target distributions [106–108], this alignment is often affected by the batch size. To stress this point, one might consider using an out-of-the-box distributional matching loss function (e.g., Maximum Mean Discrepancy [109]) to align the source and target entropy distributions, however, this approach requires large batches to obtain an effective estimation. Our approach accumulates the evidence for distribution shift in an online fashion, allowing us to use small batches, even of size one as we do in our experiments.

The work in [97] tackles the challenge of test-time adaptation where a pre-trained model must adjust to a continuously changing target domain during inference, without access to original training data. This approach, named `CoTTA`, utilizes weight-averaged and augmentation-averaged pseudo-labels to improve label accuracy and reduce error accumulation. To alleviate catastrophic forgetting, `CoTTA` intermittently reverts some neurons to their initial states. In contrast with `CoTTA`, our method operates under the assumption of no access to transformations/augmentations during testing. Employing such augmentations may further improve the performance of `POEM`, and we leave this exploration for future work.

B Learning the betting variable online

In this section we present and analyze our online approach to learn the betting variable ϵ_j, which relies on the SF-OGD algorithm [59]. The complete algorithm is presented in Algorithm 1.

In the following analysis, we assume that the betting variable that attains the fastest growth of the capital in hindsight, after observing u_τ, is in the range $[-D, D] \subset [-2, 2]$. We denote this variable by $E_\tau \in \{-D, +D\}$, which can be computed via a simple closed-form expression, given by

$$E_\tau = D \cdot \text{sign}(u_\tau - 0.5). \tag{11}$$

With this in place, we define the clip-aware loss function, which we will use to formulate the update rule for ϵ_τ that maximizes the capital:

$$L(E_\tau, \epsilon_\tau) = -\begin{cases}\log(1 + E_\tau(u_\tau - 0.5)) & \text{if } E_\tau \cdot \epsilon_\tau > 0 \text{ and } |\epsilon_\tau| > D \\ \log(1 + \epsilon_\tau(u_\tau - 0.5)) & \text{otherwise.}\end{cases} \tag{12}$$

In plain words, when ϵ_τ is out of range but has the same sign as E_τ, we clip the betting variable with the maximal value allowed (D or $-D$) that would increase the wealth. Otherwise, the loss is equal to the log of the bet, obtained with ϵ_τ.

At each step τ of the algorithm, we first predict the value of ϵ_τ, then observe u_τ, which allows us to compute E_τ. Therefore, after observing u_τ we can compute the (sub)gradient of (12):

$$\nabla_\epsilon L(E_\tau, \epsilon_\tau) = -\begin{cases}0 & \text{if } E_\tau \cdot \epsilon_\tau > 0 \text{ and } |\epsilon_\tau| > D \\ (u_\tau - 0.5)/(1 + \epsilon_\tau(u_\tau - 0.5)) & \text{otherwise.}\end{cases} \tag{13}$$

Armed with $\nabla_\epsilon L(E_\tau, \epsilon_\tau)$ we are ready to perform the SF-OGD step, formulated as:

$$\epsilon_{\tau+1} = \epsilon_\tau - \gamma \frac{\nabla_\epsilon L(E_\tau, \epsilon_\tau)}{\sqrt{\sum_{t=1}^{\tau} (\nabla_\epsilon L_t(E_t, \epsilon_t))^2}}, \tag{14}$$

where $\gamma > 0$ is a learning rate. Lemma 1 below states that ϵ_τ is indeed bounded.

Lemma 1. *The SF-OGD algorithm with a learning rate $0 < \gamma < 2 - D$ and initialization $\epsilon_1 = [-D - \gamma,\ D + \gamma]$ satisfies $\epsilon_\tau \in [-D - \gamma,\ D + \gamma]$ for all $1 \le \tau \le j$.*

Proof is in Appendix C.2. Following Lemma 1, we conclude that we must set the learning rate γ in the range $0 < \gamma < 2 - D$ to ensure that ϵ_τ is in the range $(-2, 2)$. The latter is crucial to formulate a valid betting function (4).

Building on the analysis of SF-OGD, the following proposition states that this algorithm achieves a regret bound on the loss in (12), where the regret function is defined as

$$\text{Reg}(j) = \sum_{\tau=1}^{j} L(E_\tau, \epsilon_\tau) - \sup_{\epsilon^\star \in [-D, D]} \sum_{\tau=1}^{j} L(E_\tau, \epsilon^*). \tag{15}$$

Above, $\epsilon^\star$ is the betting parameter that minimizes the loss in hindsight, over the time horizon $1 \le t \le j$.

Theorem 1. *[Theorem 4 by Orabona and pál [59]; Proposition A.2 by Bhatnagar et al. [110]] Suppose that $\epsilon^\star \in [-D, D]$. Then, SF-OGD with $E_\tau \in \{-D, D\}$ for all $1 \le \tau \le j$, learning rate $0 < \gamma < 2 - D$, and any initialization $\epsilon_1 \in [-D - \gamma, D + \gamma]$ achieves*

$$\text{Reg}(t) \le (\gamma + \frac{1}{2\gamma}(2D + \gamma)^2) \sqrt{\sum_{\tau=1}^{t} (\nabla_\epsilon L(T_\tau, \epsilon_\tau))^2} \le \mathcal{O}\left(\frac{\sqrt{t}}{2 - D - \gamma}\right), \ \ \forall 1 \le t \le j.$$

Proof is in Appendix C.3. The above result states that for any interval of size $1 \le t \le j$, the regret of SF-OGD defined in (15), is bounded by the square-root of the interval size $\sqrt{t}$, divided by the difference between the boundaries of the entire ϵ_τ domain $[-2, 2]$ and of the actual ϵ_τ domain $[D - \gamma, D + \gamma]$.

C Proofs

C.1 Proof of Proposition 1

Proof. $S_j \ge 0$ for all $j \in \mathbb{N}$ since $u_j \in [0, 1]$ and $\epsilon_j \in [-2, 2]$ by definition.
$\{S_j : j \in \mathbb{N}\}$ is a martingale under $\mathcal{H}_0$ since

$$\begin{aligned}\mathbb{E}_{\mathcal{H}_0}[S_j \mid S_1, ..., S_{j-1}] &= S_{j-1} \cdot \mathbb{E}_{\mathcal{H}_0}[b(u_j) \mid S_1, ..., S_{j-1}] \\ &= S_{j-1} \cdot (1 + \epsilon_j \cdot \mathbb{E}_{\mathcal{H}_0}[u_j - 0.5 \mid S_1, ..., S_{j-1}]) = S_{j-1},\end{aligned}$$

where the second transition holds since ϵ_j depends only on $\{S_1, ..., S_{j-1}\}$, and the latter since $\mathbb{E}_{\mathcal{H}_0}[u_j - 0.5 \mid S_1, ..., S_{j-1}] = 0$. □

C.2 Proof of Lemma 1

Proof. Leveraging the symmetry of the problem, it suffices to study the setting in which $\epsilon_\tau \geq 0$. Also, without loss of generality, we assume that $\nabla_\epsilon L_1(E_1, \epsilon_1) \neq 0$ for the first iteration of the algorithm; if this does not hold we can always remove these samples until reaching an observation with a non-zero gradient. To prove the result, we consider the following cases that can occur when optimizing (12).

- **Case 1**: $0 \leq \epsilon_\tau \leq D$.
- **Case 2**: $\epsilon_\tau > D$ and $\epsilon_\tau \cdot E_\tau \geq 0$.
- **Case 3**: $\epsilon_\tau > D$ and $\epsilon_\tau \cdot E_\tau \leq 0$.

We start by analyzing **Case 1**. Recall that $0 \leq u_\tau \leq 1$ and that $E_\tau \in \{-D, D\}$. Now, following (13) we can conclude that the gradient of the loss $L(E_\tau, \epsilon_\tau)$ is bounded $\nabla_\epsilon L(E_\tau, \epsilon_\tau) \in [-\frac{1}{2-D}, \frac{1}{2-D}]$. By recalling the update rule in (14), we get:

$$|\epsilon_{\tau+1} - \epsilon_\tau| = \gamma \left| \frac{\nabla_\epsilon L(E_\tau, \epsilon_\tau)}{\sqrt{\sum_{t=1}^{\tau} (\nabla_\epsilon L_t(E_t, \epsilon_t))^2}} \right| \leq \gamma. \tag{16}$$

With this in place, we can conclude that if $\epsilon_{\tau+1} \geq \epsilon_\tau$, then

$$\epsilon_{\tau+1} \leq \epsilon_\tau + \gamma \leq D + \gamma.$$

Otherwise, $\epsilon_{\tau+1} \leq \epsilon_\tau$, then

$$D \geq \epsilon_\tau \geq \epsilon_{\tau+1} \geq \epsilon_\tau - \gamma \geq -\gamma.$$

Above, we used the fact that $0 \leq \epsilon_\tau \leq D$ under **Case 1**. In sum, we showed that $-\gamma \leq \epsilon_{\tau+1} \leq D+\gamma$.

We now turn to analyze **Case 2**. Under the assumptions of this case $\nabla_\epsilon L(E_\tau, \epsilon_\tau) = 0$ and thus $\epsilon_{\tau+1} = \epsilon_\tau$, i.e., the value of the betting parameter will not be modified. To bound the value of $\epsilon_{\tau+1}$, we note that we can encounter $\epsilon_\tau > D$ as a result of updating the betting parameter in **Case 1**, but in this scenario, we already know that $-\gamma \leq \epsilon_\tau \leq D + \gamma$. We can also reach **Case 2** at the initialization, but then $\epsilon_1 <= D + \gamma$ and thus bounded. Lastly, another entry point to **Case 2** is from **Case 3**, however, we show below that the latter satisfies that $D - \gamma \leq \epsilon_\tau \leq D + \gamma$.

Lastly, we study **Case 3**, which can be reached from **Case 1** or **Case 2**. However, in **Case 2** $\epsilon_{\tau+1} = \epsilon_\tau$, so we can concentrate only on the scenario where **Case 3** is reached from **Case 1**, but we already showed that $\epsilon_t \leq D + \gamma$ in this case. Lastly, we can face **Case 3** when $\epsilon_1 > D$, however, it is bounded $\epsilon_1 <= D + \gamma$ by the Lemma's assumption. Hence, following (13), the gradient is bounded $\nabla_\epsilon L_t(E_t, \epsilon_t) \in [-\frac{1}{2-D-\gamma}, \frac{1}{2-D-\gamma}]$. Further, the loss can be improved only by reducing the value of ϵ_τ, and thus the SF-OGD step would result in $\epsilon_{\tau+1} \leq \epsilon_\tau$. This implies that $\epsilon_\tau - \epsilon_{\tau+1} \leq \gamma$, according to (16). In turn, by recalling that $D + \gamma \geq \epsilon_\tau > D$, we conclude that

$$D + \gamma \geq \epsilon_\tau \geq \epsilon_{\tau+1} \geq \epsilon_\tau - \gamma > D - \gamma.$$

To summarize, the analysis of the cases above indicates that $\epsilon_\tau \in [-D-\gamma,\ D+\gamma]$ for all $1 \leq \tau \leq j$, as desired. □

C.3 Proof of Theorem 1

Proof. Recall that we want to prove that

$$\text{Reg}(t) \leq (\gamma + \frac{1}{2\gamma}(2D+\gamma)^2)\sqrt{\sum_{\tau=1}^{t} (\nabla_\epsilon L(T_\tau, \epsilon_\tau))^2} \leq \mathcal{O}\left(\frac{\sqrt{t}}{2-D-\gamma}\right),\ \ \forall 1 \leq t \leq j.$$

The second inequality holds by following the proof of Lemma 1, showing that $\nabla_\epsilon L_t(E_t, \epsilon_t) \in [-\frac{1}{2-D-\gamma}, \frac{1}{2-D-\gamma}]$. The proof of the first inequality is a direct consequence of [59, Theorem 4] or [110, Proposition A.2], and thus omitted. Notable, we can directly invoke [59, Theorem 4] as (i) the loss function $L_\tau(\cdot) = L(E_\tau, \cdot)$ in (12) is convex; and (ii) the betting variable $\epsilon_\tau \in [-D-\gamma, D+\gamma]$ is bounded for all $1 \leq \tau \leq j$ due Lemma 1. □

C.4 Proof of Proposition 2

Proof. Since we assume that F_s is invertible and Z_j^t is continuous, we can conclude that F_s is a smooth bijection function. This allows us to invoke [61, Lemma 1], which states that if F_t^j is the CDF corresponding to the density function dF_t^j, and the mapping F_s is a smooth bijection, then the CDF $Q^{\text{opt}}(u) = F_t^j(F_s^{-1}(u))$ as in (9). With this in place, we can write

$$\tilde{Z}_j^t = F_s^{-1}(Q^{\text{opt}}(u_j)) = F_s^{-1}(F_t^j(F_s^{-1}(u_j))) = F_s^{-1}(F_t^j(F_s^{-1}(F_s(Z_j^t)))) = F_s^{-1}(F_t^j(Z_j^t)),$$

where the second equality holds due [61, Lemma 1] and the third equality stems from the definition of u_j, being $u_j = F_s(Z_j^t)$. We conclude the proof by observing that the mapping $\tilde{Z}_j^t = F_s^{-1}(F_t^j(Z_j^t))$ is the optimal transport map from the target to the source distribution w.r.t. the Wasserstein distance. This is because Z_j^t is a continuous, one-dimensional random variable, with an invertible CDF F_s [111]. □

D Using martingales to detect distribution shifts

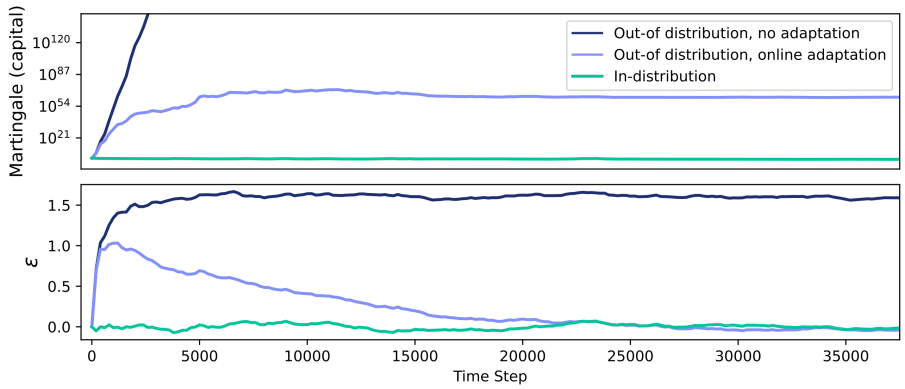

Figure 4: **Martingale behaviour with and without adaptation and on in-distribution data.** Visualization of three scenarios: (1) out-of-distribution data (ImageNet-C, brightness level 1) without adaptation, (2) the same out-of-distribution data with online adaptation, and (3) in-distribution data (ImageNet) all on ResNet50. The top panel shows the martingale value, that is, the accumulated capital (in log scale) over time, while the bottom panel shows the corresponding betting variable ϵ.

Recall that we established in Proposition 1 the validity of the martingale defined in (4). In this section, we describe how a valid test martingale can be used to detect distribution shifts with a type-I error guarantee. This this end, consider a test martingale sequence denoted by $\{S_j : j \in \mathbb{N}_0\}$ under the null hypothesis $\mathcal{H}_0$ of no distribution shift (3). Ville's inequality [112] plays a crucial role in bounding the probability of this martingale exceeding a specific threshold under $\mathcal{H}_0$. Specifically, for any value of α between 0 and 1, Ville's inequality states the following:

$$\mathbb{P}_{\mathcal{H}_0}\left(\exists j \geq 1 : S_j \geq \frac{1}{\alpha}\right) \leq \alpha \mathbb{E}_{\mathcal{H}_0}[S_0] = \alpha. \tag{17}$$

Suppose we set a significance level $\alpha = 0.01$. The above inequality states that under the assumption of no distribution shift, the martingale's value will exceed a threshold of $1/\alpha$ (in this case, 100) with a probability of at most $\alpha = 0.01$. This bound on the probability allows us to simultaneously control the type-I error across all time steps. This implies that we can declare that we face a distribution shift if the martingale value pass the threshold $1/\alpha$. For instance, if we set the threshold to 100 or higher, the type-I error is guaranteed to be less than or equal to 0.01.

Figure 4 top panel offers a visual representation of the martingale's behavior under different scenarios; see Appendix F for implementation details. As discussed above, under the null hypothesis of no distribution shift, the martingale is expected to remain lower than 1 with high probability. This is precisely what we observe in the in-distribution data scenario—the martingale value remains under

1. Here, we consider the ImageNet dataset, where we applied a pre-trained ResNet50 model on the ImageNet validation set.

In contrast, when the pre-trained model is applied to corrupted, out-of-distribution data (ImageNet-C), the martingale value deviates significantly from 1, reaching levels of up to 10^{200}. This substantial increase validates the presence of a distribution shift. Interestingly, Figure 4 also reveals the effectiveness of our online adaptation. The adaptation process gradually limits the martingale value from growing compared to the non-adaptation case. Observe how the martingale of the adapted model eventually converges to a plateau. This visually demonstrates the success of adapting the model to the new distribution, making the target entropies statistically indistinguishable from the source data.

E Algorithms

In this section, we present a series of algorithms essential to understanding and implementing our proposed methods. It is important to note that throughout these algorithms, we do not explicitly state each instance where lists or variables are updated; however, such updates are implicitly understood to occur during computations.

We use the "**Assume**" directive in our algorithms to outline which variables are accessible as global variables. These global variables are updated as part of the algorithms' operations but are not repeatedly declared within each algorithmic step. This approach is chosen to streamline the presentation and focus on the algorithmic logic rather than the mechanics of data handling.

Algorithm 1 SF-OGD Step

Require: $u_j \in [0, 1]$
Assume:
ϵ_j: last betting parameter's value
$\{\nabla_\epsilon L(E_0, \epsilon_0), ..., \nabla_\epsilon L(E_{j-1}, \epsilon_{j-1})\}$: past gradient values of the log loss from (12)
D: betting variable clip value
γ: SF-OGD learning rate

$E_j \leftarrow D \cdot \text{sign}(u_j - 0.5)$ ▷ Following (11)
if $E_j \cdot \epsilon_j > 0$ and $|\epsilon_j| > D$ **then**
    $\nabla_\epsilon L(E_j, \epsilon_j) \leftarrow 0$ ▷ Following (13)
else
    $\nabla_\epsilon L(E_j, \epsilon_j) \leftarrow -\frac{u_j - 0.5}{1 + \epsilon_j(u_j - 0.5)}$ ▷ Following (13)
end if
$\epsilon_{j+1} \leftarrow \epsilon_j - \gamma \frac{\nabla_\epsilon L(E_j, \epsilon_j)}{\sqrt{\sum_{t=1}^{j} (\nabla_\epsilon L_t(E_t, \epsilon_t))^2}}$ ▷ Following (14)
return ϵ_{j+1}

Algorithm 2 Protected Online Entropy Matching (POEM)

Require:
$\mathcal{D}^s = \{X_j^s\}_{j=1}^n$: holdout data from source distribution
$f_{\hat{\theta}}$: pretrained model
D: last betting parameter's value
γ: SF-OGD learning rate
η: model learning rate
λ: entropy filter threshold, see (10)

Init: $\epsilon_1 = 0$, $\omega_1 \leftarrow \mathbf{0}$ ▷ We update only the network's norm. layers
Compute empirical CDF function of source entropies
for X_i^s in $\mathcal{D}^s$ **do**
    $Z_i^s \leftarrow \ell^{\text{ent}}(f_{\hat{\theta}}(X_i^s))$
end for
Define: $\hat{F}_s(z) = \frac{1}{n}\sum_{i=1}^n \mathbb{1}\{Z_i^s \leq z\}$ ▷ Empirical CDF function
Online adaptation
$j \leftarrow 1$
while Get a test sample X_j^t **do**
    $Z_j^t \leftarrow \ell^{\text{ent}}(f_{\hat{\theta}+\omega_j}(X_j^t))$ ▷ Compute test entropy
    $u_j \leftarrow \hat{F}_s(Z_j^t)$ ▷ Apply probability integral transform
    $b_j = 1 + \epsilon_j \cdot (u_j - 0.5)$ ▷ Place a bet against the null (3)
    $\tilde{u}_j \leftarrow \frac{1}{2}\epsilon_j u_j^2 + u_j \cdot (1 - \frac{\epsilon_j}{2})$ ▷ Adapt u_j according to (7)
    $\tilde{Z}_j \leftarrow \{\min z : \hat{F}_s(z) \geq \tilde{u}_j\}_{i=1}^n$ ▷ Transport to source domain, $\hat{F}_s^{-1}(\tilde{u}_j)$
    $\omega_{j+1} \leftarrow \omega_j - \eta\nabla_\omega \ell_\lambda^{\text{match++}}(Z_j^t, \tilde{Z}_j)$ ▷ Update norm. layers' params. according to (10)
    $\epsilon_{j+1} \leftarrow$ Alg. 1 (u_j) ▷ Update the betting variable
    $j \leftarrow j + 1$
end while

F Supplementary experiments

F.1 Synthetic experiment

To evaluate the methods, we generate a synthetic dataset by generating two sets of points, each containing $2,500$ samples. Specifically, samples for the first set (class -1) are drawn from a normal distribution $\mathcal{N}(-1, 1)$, while samples for the second set (class $+1$) are drawn from $\mathcal{N}(1, 1)$. We use these $5,000$ samples to calculate the source CDF. Note that we do not use a training set, as we initialized the pre-trained model's parameter ω to the optimal value, which is $\omega = 0$.

We then create two test sets:

- **In-distribution test data**, which consists of 10,000 samples per class, following the source distribution.
- **Out-of-distribution test data**, which consists of 10,000 samples per class, but with shifted distributions—the class -1 samples are drawn from $\mathcal{N}(0, 1)$ and the class $+1$ samples are drawn from $\mathcal{N}(2, 1)$. This represents a distributional shift of 1 unit in X.

We defined our model parameter as $\omega \in \mathbb{R}$. Two types of loss functions were employed:

- Entropy loss ℓ^{ent} (1), computed by evaluating the entropy of each point's prediction given the model parameter ω.
- Matching loss ℓ^{match} (2), computed using an optimal transport mapping, i.e., $F_s^{-1} \circ F_t$, to match the new set of target entropies derived from f_ω to the source entropies obtained from the holdout data.

Hyperparameters and training scheme The optimization of each method was executed over 200 steps using a fixed learning rate of 5. Although lower learning rates were also effective, a higher

learning rate was chosen to accelerate the demonstration. The optimization was performed with a batch size of 64. Each method's performance was evaluated at the end of this process. We clipped entropy minimization convergence if it went outside the bounds of our plot. In all experiments conducted in this paper, we choose the following set of hyperparameters, defined in Algorithm 1:

- $D = 1.8$.
- $\gamma = \frac{1}{8 \cdot \sqrt{3}} \approx 0.0722$.

F.2 ImageNet-C experiments

Models Our experiments utilize two pre-trained architectures: a Vision Transformer (ViT) with layer normalization (LN) and a ResNet50 with group normalization (GN). Both are pre-trained models from the `timm` library. We calibrate both models using temperature scaling, setting the temperature of ResNet50 to $T = 0.90$ and $T = 1.025$ for ViT. For `POEM` specifically, we implement an action delay of 100 examples throughout the experiments in this paper. This delay allows the monitoring component of `POEM` to accumulate sufficient evidence before updating the model parameters, mitigating the influence of potentially noisy initial data.

Data We randomly sample 25% of the examples from ImageNet validation set as an unlabelled holdout set. The corresponding corrupted examples are excluded from ImageNet-C to maintain the validity of the holdout data. All methods are evaluated only on the remaining 75% samples. Each experiment is repeated 10 times with different holdout data splits and data selections, which is particularly crucial for experiments with small subsets of examples, such as the continual shifts experiments. Specifically for ViT, we modify the default preprocessing transforms offered by `SAR`, to the one used by the model during its training (see `https://huggingface.co/timm/vit_base_patch16_224.augreg2_in21k_ft_in1k` for the exact details). This adjustment is crucial to ensure proper estimation of the source CDF as well as ensure the model's performance on in-distribution data adheres to the one reported in Hugging Face.

Code, hyperparameters, and learning scheme We use the `SAR` [43] repository (available at `https://github.com/mr-eggplant/SAR`) for the ImageNet and ImageNet-C experiments. To ensure consistency with prior works, we adopt the hyperparameters, optimizers, and procedures provided within the `SAR` repository. This ensures that all baseline methods as well as `POEM` are run with the exact same settings. We also compare our method with `COTTA`[97] using the code provided by the authors, available at `https://github.com/qinenergy/cotta`. In more detail:

- For all methods except `COTTA`:
  - The learning rate (η in Algorithm 2) calculation follows these formulas:
    - ViT: learning rate $= \left(\frac{0.001}{64}\right) \times$ batch size
    - ResNet50: learning rate $= \left(\frac{0.00025}{64}\right) \times$ batch size $\times\, 2$
  - We use `SGD` optimizer with momentum of 0.9 for self-training.
  - We use $\lambda = 0.40 \cdot \log(1000)$ (denoted by E_0 in [42, 43]).
- For `COTTA`:
  - We tune the learning-rate search for each model (ResNet and ViT), ranging from $\frac{0.001}{64} \cdot i$ where $i \in [0.5, 1, 2, 4, 8]$, and select the best-performing value for each model on the continual setting with a corruption segment size of $1,000$. This process resulted in the following learning rate values.
    - ViT learning rate $= \left(\frac{0.001}{64}\right) \times 4$
    - ResNet50 learning rate $= \left(\frac{0.001}{64}\right) \times 2$
  - We use `Adam` optimizer with $\beta = (0.9, 0.999)$ and weight decay 0 for self-training, as employed in the original work.

A batch size of 1 is consistently used throughout all of the experiments, and, as mentioned in Appendix F, we use $D = 1.8$ and $\gamma = \frac{1}{8 \cdot \sqrt{3}} \approx 0.0722$ for our monitoring algorithm (see Algorithm 1). Lastly, since the empirical CDF in Algorithm 2 (line 6) is calculated from a finite set of data points, it inherently creates a step function. In practice, we use linear interpolation to create a continuous function for better operation.

Hardware All experiments are conducted on our local server, equipped with 16 `NVIDIA A40 GPU - 49GB` GPUs, 192 `Intel(R) Xeon(R) Gold 6336Y` CPUs, and 1TB of RAM memory. Each experiment uses a single GPU and 8 CPUs.

F.2.1 Additional experiments: continual shifts

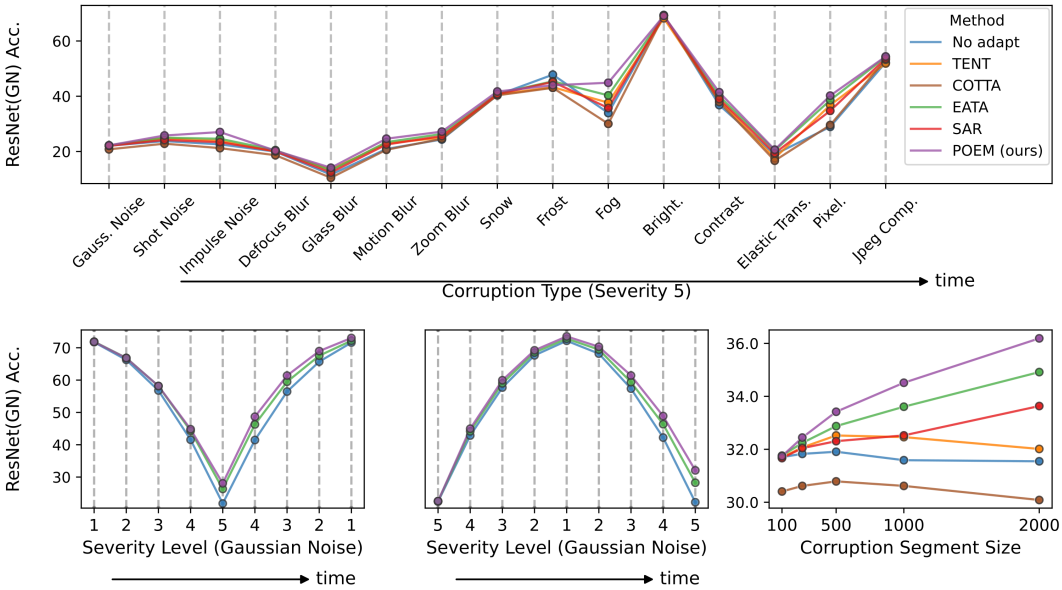

Figure 5: **Continual test-time adaptation on ImageNet-C with a ResNet model**. **Top:** Per-corruption accuracy with a corruption segment size of 1,000 examples. Results are obtained over 10 independent trials; error bars are tiny. **Bottom left:** Severity shift—low (1) to high (5) and back to low. **Bottom center:** Severity shift—high (5) to low (1) and back to high. **Bottom right:** Mean accuracy under continual corruptions as a function of the corruption segment size.

Supplementary details for the continual shifts experiments presented in the main manuscript For the corruption shift experiment, we used corruption segments sizes in the range of 100, 250, 500, 1000, and 2000. Apart from the bottom right panels of Figures 2 and 5, the results are obtained using corruption segment size of 1,000 exclusively.

Experiments with a ResNet (GN) model Herein, we repeat the same experiments from Section 4 of the main manuscript to evaluate our proposed method within a continual shift setting, but now focus on a ResNet50 (GN) model. The results are summarized in Figure 5, showing a similar trend to that obtained with the ViT model, albeit accuracies that are closer to the baseline methods. We note that the baseline accuracy of ResNet is far below that of ViT. The bottom right panel of Figure 5 demonstrates `POEM`'s efficiency in adapting to fast-changing distributions with only a few examples. While the advantage of `POEM` is less clear with small corruption segments in ResNet, it becomes evident as early as segment size 500. Observe the bottom right panel of Figure 5. At corruption segment size of 2,000, `POEM` surpasses the best baseline method's accuracy (`EATA`) by 1.27%. `POEM` also achieves 4.64% increase compared to the original model (no adaptation).

F.2.2 Additional experiments: single shift

Table 1 summarizes baseline methods' accuracy on ImageNet-C for ViT and ResNet models. Table 2 offers a more detailed breakdown of the results in Table 1, showing accuracy for each corruption type.

F.2.3 Additional experiments: in-distribution and the behavior of ϵ

Even though adapting on unseen in-distribution data from ImageNet attains similar accuracies for all baseline methods compared to the no-adapt approach (Table 3), `POEM` maintains this accuracy with the most minimal change to the original model's parameters ω and virtually unchanged ECE. This

Table 1: **Single shift test-time adaptation**. Accuracy evaluated on ImageNet-C, averaged over all 15 corruptions of severity level 5. Detailed results are in Table 2. We attribute COTTA's inferior performance to its learning rate being tuned for the continual setting; see Appendix F.2 for details.

Method	ResNet50 (GN)	ViT (LN)
No adapt	31.44	51.65
TENT	25.46	62.21
COTTA	23.90	55.34
EATA	38.63	64.14
SAR	36.01	64.03
POEM (ours)	**38.90**	**67.36**

Table 2: **Summary of performance metrics of adaptation methods on the ImageNet-C dataset**, evaluated for each type of corruption at severity level 5. The results are evaluated on 10 independent experiments, conducted for each method and corruption type; standard error is presented.

Corruption / Method		Gauss. Noise	Shot Noise	Impulse Noise	Defocus Blur	Glass Blur	Motion Blur	Zoom Blur	Snow	Frost	Fog	Bright.	Contrast	Elastic Trans.	Pixel.	Jpeg Comp.
ResNet50 (GN)	No adapt	22.11	23.05	22.04	**19.82**	11.44	21.44	25.01	40.30	**47.00**	33.98	68.82	36.26	18.55	29.24	52.59
	TENT	$12.22_{\pm 1.5}$	$14.50_{\pm 1.4}$	$15.80_{\pm 1.5}$	$14.49_{\pm 0.1}$	$6.36_{\pm 0.9}$	$20.62_{\pm 0.1}$	$20.59_{\pm 0.3}$	$22.37_{\pm 0.4}$	$27.90_{\pm 0.4}$	$1.88_{\pm 0.0}$	$70.26_{\pm 0.0}$	$42.65_{\pm 0.1}$	$8.10_{\pm 0.1}$	$49.65_{\pm 0.1}$	$54.55_{\pm 0.1}$
	COTTA	$5.72_{\pm 0.1}$	$7.60_{\pm 0.1}$	$5.02_{\pm 0.1}$	$10.72_{\pm 0.5}$	$3.52_{\pm 0.0}$	$20.02_{\pm 0.2}$	$24.65_{\pm 0.1}$	$35.78_{\pm 0.2}$	$42.09_{\pm 0.1}$	$4.45_{\pm 0.1}$	$\mathbf{72.01_{\pm 0.1}}$	$34.92_{\pm 0.5}$	$16.03_{\pm 0.1}$	$19.97_{\pm 0.8}$	$55.96_{\pm 0.1}$
	EATA	$31.22_{\pm 0.1}$	$33.07_{\pm 0.1}$	$32.51_{\pm 0.1}$	$18.41_{\pm 0.1}$	$18.61_{\pm 0.1}$	$29.41_{\pm 0.1}$	$30.11_{\pm 0.1}$	$\mathbf{45.11_{\pm 0.1}}$	$45.02_{\pm 0.1}$	$\mathbf{47.30_{\pm 0.1}}$	$70.76_{\pm 0.0}$	$45.06_{\pm 0.1}$	$\mathbf{28.18_{\pm 0.2}}$	$48.66_{\pm 0.1}$	$55.95_{\pm 0.1}$
	SAR	$31.37_{\pm 0.1}$	$33.88_{\pm 0.1}$	$32.32_{\pm 0.1}$	$18.87_{\pm 0.1}$	$17.72_{\pm 0.1}$	$30.34_{\pm 0.0}$	$32.00_{\pm 0.1}$	$40.99_{\pm 0.5}$	$\mathbf{45.31_{\pm 0.0}}$	$21.33_{\pm 5.2}$	$\mathbf{72.01_{\pm 0.1}}$	$45.69_{\pm 0.1}$	$11.48_{\pm 0.1}$	$50.43_{\pm 0.1}$	$56.44_{\pm 0.0}$
	POEM (ours)	$\mathbf{39.90_{\pm 0.1}}$	$\mathbf{42.16_{\pm 0.1}}$	$\mathbf{41.03_{\pm 0.1}}$	$18.86_{\pm 0.4}$	$\mathbf{22.14_{\pm 0.1}}$	$\mathbf{38.01_{\pm 0.1}}$	$\mathbf{36.16_{\pm 2.3}}$	$21.59_{\pm 0.7}$	$42.72_{\pm 2.6}$	$35.63_{\pm 7.5}$	$71.94_{\pm 0.1}$	$\mathbf{50.43_{\pm 0.1}}$	$9.06_{\pm 0.2}$	$\mathbf{55.85_{\pm 0.1}}$	$\mathbf{57.94_{\pm 0.0}}$
ViT (LN)	No adapt	49.73	50.27	50.03	42.72	34.43	50.57	44.79	56.61	52.31	56.63	75.86	31.93	46.89	65.53	66.39
	TENT	$57.70_{\pm 0.0}$	$58.80_{\pm 0.0}$	$58.67_{\pm 0.1}$	$57.28_{\pm 0.0}$	$53.18_{\pm 0.1}$	$60.47_{\pm 0.1}$	$56.87_{\pm 0.1}$	$64.93_{\pm 0.1}$	$52.78_{\pm 2.0}$	$68.39_{\pm 0.1}$	$78.42_{\pm 0.0}$	$61.86_{\pm 0.1}$	$61.31_{\pm 0.0}$	$72.16_{\pm 0.0}$	$70.29_{\pm 0.0}$
	COTTA	$53.17_{\pm 0.4}$	$54.72_{\pm 0.3}$	$54.84_{\pm 0.3}$	$45.08_{\pm 1.1}$	$44.37_{\pm 0.3}$	$59.42_{\pm 0.2}$	$53.12_{\pm 0.1}$	$57.80_{\pm 0.9}$	$49.44_{\pm 0.5}$	$56.95_{\pm 1.1}$	$79.39_{\pm 0.0}$	$17.30_{\pm 1.5}$	$59.92_{\pm 0.2}$	$73.16_{\pm 0.2}$	$71.46_{\pm 0.1}$
	EATA	$58.73_{\pm 0.0}$	$60.03_{\pm 0.1}$	$59.88_{\pm 0.1}$	$58.04_{\pm 0.0}$	$54.86_{\pm 0.1}$	$61.14_{\pm 0.0}$	$57.68_{\pm 0.1}$	$66.49_{\pm 0.1}$	$65.35_{\pm 0.0}$	$69.08_{\pm 0.1}$	$79.30_{\pm 0.0}$	$62.28_{\pm 0.1}$	$63.88_{\pm 0.1}$	$73.31_{\pm 0.1}$	$72.02_{\pm 0.0}$
	SAR	$58.77_{\pm 0.0}$	$59.97_{\pm 0.0}$	$59.85_{\pm 0.1}$	$57.89_{\pm 0.0}$	$54.32_{\pm 0.1}$	$61.58_{\pm 0.0}$	$58.19_{\pm 0.1}$	$66.61_{\pm 0.1}$	$64.92_{\pm 0.1}$	$69.07_{\pm 0.1}$	$78.70_{\pm 0.1}$	$61.55_{\pm 0.1}$	$64.22_{\pm 0.1}$	$73.31_{\pm 0.0}$	$71.53_{\pm 0.1}$
	POEM (ours)	$\mathbf{60.94_{\pm 0.0}}$	$\mathbf{62.60_{\pm 0.0}}$	$\mathbf{62.47_{\pm 0.1}}$	$\mathbf{60.08_{\pm 0.1}}$	$\mathbf{60.66_{\pm 0.1}}$	$\mathbf{65.23_{\pm 0.1}}$	$\mathbf{63.41_{\pm 0.0}}$	$\mathbf{70.05_{\pm 0.0}}$	$\mathbf{68.57_{\pm 0.1}}$	$\mathbf{73.39_{\pm 0.1}}$	$\mathbf{79.51_{\pm 0.0}}$	$\mathbf{63.99_{\pm 0.4}}$	$\mathbf{70.60_{\pm 0.0}}$	$\mathbf{75.45_{\pm 0.0}}$	$\mathbf{73.43_{\pm 0.1}}$

is demonstrated in the left panels of Figure 3 and Figure 6. Specifically, as seen in Table 3, $\|\omega\|_F^2$ for POEM is merely 0.17, significantly lower than other methods on ResNet50—over 30 times less change than the closest baseline method. This minimal change demonstrates POEM's capability for controlled adaptation.

Figure 7 further demonstrates the "no-harm" effect of POEM under in-distribution test data. The left panel shows the empirical CDF of the entropy values $\hat{F}_t$ of all baseline methods, indicating that these lead to overconfident predictions. This is in contrast to POEM, whose estimated target CDF closely aligns with the source distribution. Such a minimal deviation from the source distribution aligns with our previous findings—POEM tends to keep the model parameters intact when adaptation is unnecessary.

When applying the model to out-of-distribution data (right panel of Figure 7), the unadapted model appears slightly under-confident, as indicated by its corresponding CDF $\hat{F}_t$ being lower than $\hat{F}_s$. Here, POEM effectively restores the model's original confidence by adjusting its estimated CDF to closely match the source CDF. SAR achieves comparable results to POEM, but through a different strategy. It employs a restart mechanism that acts as a safeguard, activated when the entropy of the adapted model drops below a specific exponential moving average (EMA) threshold.

F.2.4 Ablation study

We assess the impact of $\ell^{\text{match++}}$ (10) compared to ℓ^{match} (2) on adaptation accuracy through isolated analysis. Table 4 shows that our basic match loss ℓ^{match} improves the test accuracy over the no-adapt baseline, even without filtering and weighting. This underscores the core strength of our approach. The enhanced $\ell^{\text{match++}}$ that incorporates sample filtering and weighting mechanisms further boosts performance.

F.3 CIFAR10-C and CIFAR100-C experiments

Data CIFAR10-C and CIFAR100-C datasets are extensions of the original CIFAR10 and CIFAR100 test sets. These datasets consist of 15 different corrupted versions of the original CIFAR test images, with each corruption applied at 5 severity levels, mirroring the structure of ImageNet-C. In our experimental setup, we randomly select 25% of the examples from original CIFAR test set to create

Table 3: Results of adaptation on in-distribution ImageNet data.

Model	Method	Top-1 acc.	Top-5 acc.	Empirical cal. err.	$\|\omega\|_F^2$
ResNet50 (GN)	No adapt	$79.95_{\pm 0.04}$	$94.94_{\pm 0.02}$	$0.0243_{\pm 0.00}$	$0.00_{\pm 0.00}$
	TENT	$79.63_{\pm 0.05}$	$94.78_{\pm 0.02}$	$0.0894_{\pm 0.00}$	$5.90_{\pm 0.01}$
	COTTA	$79.87_{\pm 0.04}$	$94.83_{\pm 0.02}$	$0.0341_{\pm 0.00}$	$29.08_{\pm 0.02}$
	EATA	$79.91_{\pm 0.04}$	$94.89_{\pm 0.02}$	$0.0709_{\pm 0.00}$	$10.53_{\pm 0.05}$
	SAR	$79.97_{\pm 0.05}$	$94.93_{\pm 0.02}$	$0.0250_{\pm 0.00}$	$8.14_{\pm 0.01}$
	POEM (ours)	$79.95_{\pm 0.04}$	$94.93_{\pm 0.02}$	$\mathbf{0.0243_{\pm 0.00}}$	$\mathbf{0.17_{\pm 0.03}}$
ViT (LN)	No adapt	$84.52_{\pm 0.03}$	$97.30_{\pm 0.02}$	$0.0099_{\pm 0.00}$	$0.00_{\pm 0.00}$
	TENT	$84.42_{\pm 0.04}$	$97.30_{\pm 0.02}$	$0.0135_{\pm 0.00}$	$4.35_{\pm 0.00}$
	COTTA	$84.47_{\pm 0.04}$	$97.35_{\pm 0.01}$	$0.0209_{\pm 0.00}$	$13.75_{\pm 0.02}$
	EATA	$84.57_{\pm 0.04}$	$97.35_{\pm 0.02}$	$0.0141_{\pm 0.00}$	$6.70_{\pm 0.05}$
	SAR	$84.52_{\pm 0.03}$	$97.32_{\pm 0.02}$	$0.0101_{\pm 0.00}$	$0.86_{\pm 0.20}$
	POEM (ours)	$84.48_{\pm 0.03}$	$97.29_{\pm 0.02}$	$\mathbf{0.0098_{\pm 0.00}}$	$\mathbf{0.32_{\pm 0.05}}$

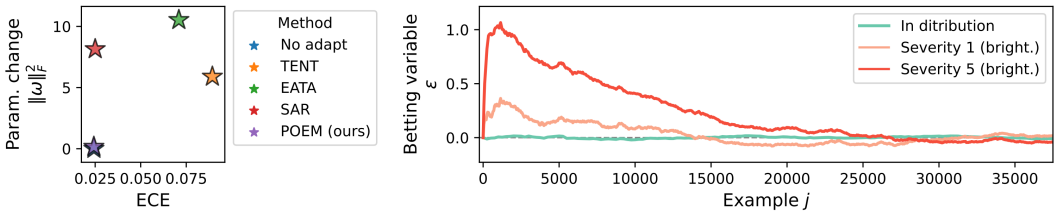

Figure 6: **In-distribution experiment on ImageNet (left panel)**: calibration error (ECE [100]) versus $\|\omega\|_F^2$—a metric that evaluates the classifier's parameters deviation from the original ResNet50 model. Lower values on both axes are better. Results are averaged across 10 independent trials; standard errors and accuracy of each method are reported in Table 3 in the appendix. **The behavior of the betting parameter (right panel)**: the value of ϵ is presented as a function of time for both in- and out-of-distribution experiments (a single shift, two severity levels).

unlabelled holdout sets of sizes 2,500 images. To preserve the integrity of the holdout set, we exclud the corresponding corrupted examples from CIFAR10-C and CIFAR100-C, respectively. All adaptation methods were applied on the remaining 75% of the data, ensuring consistency across approaches, regardless of their holdout set requirements. We conduct each experiment for 10 independent trials.

Model Our experiments utilize a pre-trained ResNet32 model with batch normalization (BN), obtained from `torch-hub` and available at `https://github.com/chenyaofo/pytorch-cifar-models`.

Methods, code, hyperparameters, and learning scheme The pre-trained ResNet32 architecture includes batch normalization (BN) layers, which forces us to use a batch-size of 4 during self-training. This differs from the batch-size we used in the ImageNet experiments, which was equal to 1. In what follows, we compare `POEM` to `SAR`, `EATA`, and `TENT` only. We do not conduct experiments with `COTTA`, as our ImageNet experiments showed that `COTTA` performs inferiorly when the self-training batch size is small. To ensure fair comparison, we perform a grid search for choosing the optimal learning

Table 4: **Ablation study: the effect of ℓ^{match} compared to $\ell^{\text{match++}}$ on the accuracy of** `POEM`. Results are presented for ImageNet-C and averaged over all 15 corruptions of severity level 5.

Method	$\ell^{\text{match++}}$	ResNet50 (GN)	ViT (LN)
No adapt	n/a	31.46	51.65
`POEM`	✗	32.49	60.64
	✓	38.90	67.36

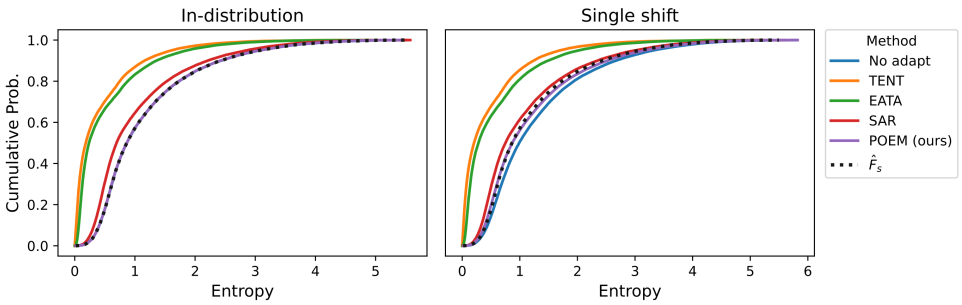

Figure 7: **Empirical test entropy CDF of each adaptation method, applied to in- and out-of-distribution ImageNet data.** The dotted black line represents the source CDF $\hat{F}_s$ obtained by applying the original ResNet50 model on test images from the source domain. The **left** panel shows how self-training on the validation set of ImageNet (in-distribution data) affects the entropy distribution of the model. The **right** panel repeats the experiment on out-of-distribution data from ImageNet-C with brightness corruption of severity level 1.

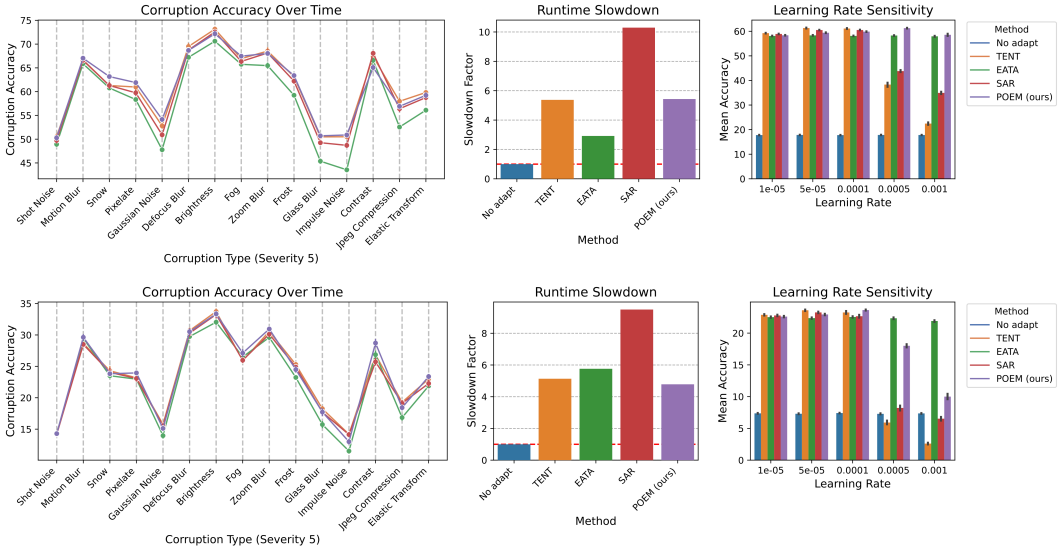

Figure 8: **Short-term adaptation performance on a test set of about 15,360 samples from CIFAR10-C (top) and CIFAR100-C (bottom) using a ResNet (BN) model. Left:** Continual test-time adaptation performance showing per-corruption accuracy with a corruption segment size of 1,024 examples of severity level 5. Results are averaged over 10 independent trials with error bars indicated. The 'no adapt' baseline is not displayed as its mean accuracy is significantly lower than other methods (see the right panel); we omitted this method to enhance visualization of differences between the adaptation techniques. **Middle:** Runtime slowdown, defined as the runtime of test-time adaptation divided by the runtime of the source (no-adapt) model; lower is better. **Right:** Learning rate sensitivity study, showing mean performance across the continual experiment. The results in the left panel are obtained with the best learning rate for each method.

rate η for each method. The details of this sensitivity analysis are presented in the following section. In the case of `POEM`, we retained the hyperparameters of the monitoring tool as outlined in Section F.

F.3.1 Continual shifts experiments

Short-term adaptation performance We focus on the continual setting where the corruption type is changing over time, akin to our ImageNet experiments. Each corruption type includes $1,024$ samples, resulting in a test set of approximately $15 \cdot 1,024 \approx 15,000$ samples. The results are summarized in Figure 8. Following that figure, one can see that our method is competitive and even

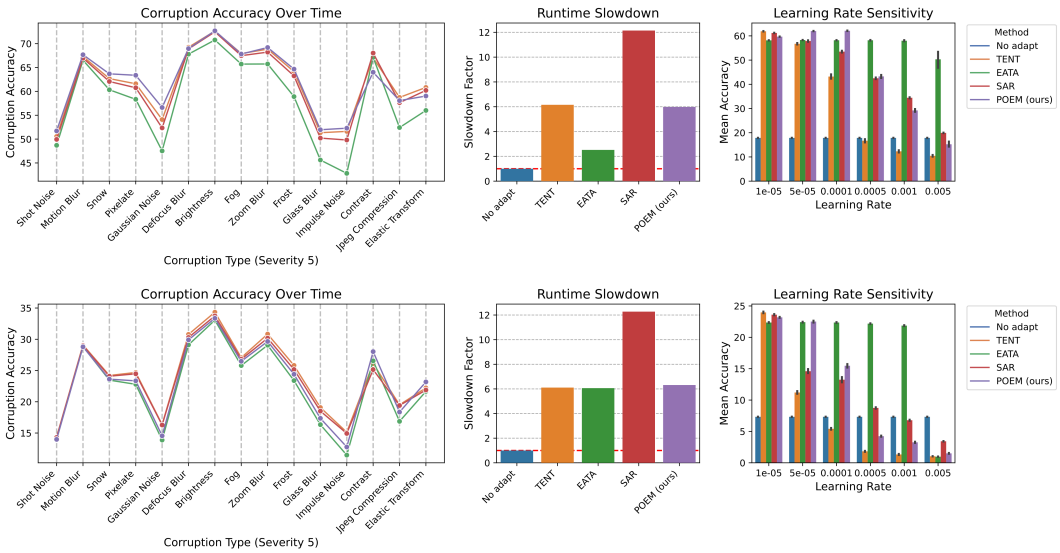

Figure 9: **Long-term adaptation performance on a test set of 112,500 samples from CIFAR10-C (top) and CIFAR100-C (bottom) using a ResNet (BN) model.** Continual test-time adaptation is applied with a corruption segment size of 7,500 examples of severity level 5. The other details are as in Figure 8.

outperforms baseline methods in terms of accuracy. Runtime comparisons (relative to the no-adapt model) are also presented, demonstrating that our method's complexity is similar to `TENT` and `EATA`, and lower than `SAR`. Additionally, the sensitivity study for the learning rate parameter η reveals our method's robustness to this hyperparameter, particularly when compared to `SAR` and `TENT`.

Long-term adaptation performance Here, we conduct a similar experiment, but on a larger test set containing 112,500 corrupted samples (15 versions of 7,500 images). The results, summarized in Figure 9, show that our proposed method is competitive with the baseline methods in terms of adaptation accuracy. Notably, our method's runtime is twice as fast as `SAR` and comparable to `EATA` and `TENT`. Unlike `SAR`, we do not employ a model-reset mechanism, nor do we use an anti-forgetting loss like `EATA`—yet our approach demonstrates robustness over the long term. In contrast, the right panel of Figure 9 reveals that `TENT` is highly sensitive to the choice of learning rate.

F.4 Office-Home experiments

Data The OfficeHome dataset consists of images from 4 domains: "Art", "Clipart", "Product", and "Real World". It contains a total of 15,588 images across 65 object categories. The "Art" domain has 2,427 images, "Clipart" has 4,365 images, "Product" has 4,439 images, and "Real World" has 4,357 images. We focus on adaptation from the "Real World" domain to the "Art", "Clipart", and "Product" domains. We chose this setup over a continual setting as it is deemed more natural for this dataset. Given the lack of a predefined data structure, we split the dataset into an 80% training set from the "Real World" samples, with the remainder serving as validation and holdout sets for our method and `EATA`.

Methods, model, hyperparameters, and learning scheme We fine-tune the last layer of the ResNet50 with Group Normalization (GN) previously used in the ImageNet experiments. We fit the model on 80% of the "Real World" examples for 25 epochs, with the best model saved based on performance on the remaining 20%. We use Adam optimizer with default PyTorch hyperparameters and set the learning rate to $5 \cdot 10^{-5}$. Similar to the CIFAR experiments, we compare `POEM` to `SAR`, `EATA`, and `TENT` only; our ImageNet experiments showed that `COTTA` performs inferiorly when the self-training batch size is 1, which is used in this experiment as well. Learning rates are tuned for each method using a predefined grid, ensuring a fair comparison, similar to our CIFAR experiments. The hyperparameters of `POEM`'s monitoring tool are as specified in Section F.

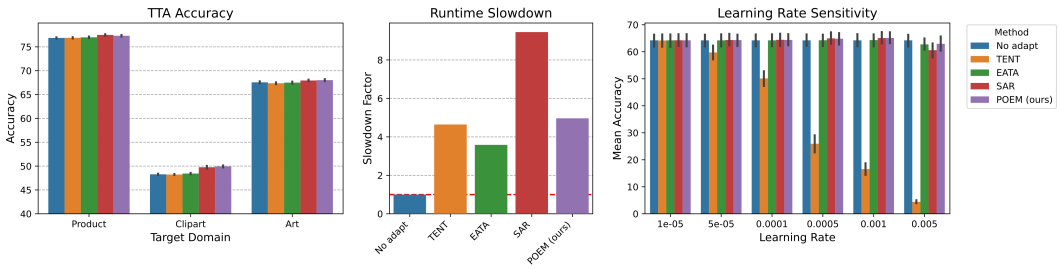

Figure 10: **Performance analysis of test-time adaptation methods on the Office-Home dataset using a ResNet (GN) model, pre-trained on ImageNet and fine-tuned on the "Real World" source domain. Left:** Test-time accuracy for adaptation from the source domain to three distinct target domains ("Art", "Clipart", and "Product"). Results are evaluated on the complete test dataset of each target domain and averaged across 10 independent trials, with error bars indicated. **Middle:** Runtime slowdown comparison. **Right:** Learning rate sensitivity study, displaying the average accuracy across all three target domains. The results in the left panel are obtained using the best learning rate for each method.

F.4.1 Single domain adaptation experiments

Test-time adaptation is applied to the entire test data of each target domain ("Art", "Clipart" and "Product"). Results are summarized in Figure 10. Overall, all the methods demonstrate modest accuracy gains compared to the 'no-adapt' case. Our proposed method `POEM` slightly outperforms `TENT` and `EATA` in terms of accuracy, while achieving results comparable to `SAR`. In terms of computational efficiency, our method's runtime is on par with `TENT` and `EATA`, and notably faster than `SAR`. Regarding sensitivity to the choice of the learning rate, our approach displays superior robustness compared to `TENT` and `SAR`, and a similar robustness to that of `EATA`.

G Future Directions

In future work, we plan to complement our empirical findings with a theoretical analysis. Our goal is to rigorously determine when entropy matching is superior to entropy minimization, thereby uncovering the theoretical benefits and limitations of our approach.

Another future direction is to support `POEM` with the ability to handle label shift at test time. This challenge is exemplified by scenarios where the source domain has a balanced label distribution, but the test domain becomes unbalanced. In such cases, our current monitoring tool might detect this label shift and trigger unnecessary adaptation in the absence of covariate shift. This underscores the need for a monitoring tool that remains invariant to label shifts. To address this challenge, one may consider two potential approaches. The first builds on ideas from [113], particularly their prediction-balanced reservoir sampling technique. This method can be used to approximately simulate an i.i.d. data stream from a non-i.i.d. stream in a class-balanced manner, potentially reducing our martingale process's sensitivity to label shifts. The second approach may involve the use of a weighted source CDF instead of the standard source CDF, with weights corresponding to the likelihood ratio $P^t(Y)/P^s(Y)$. This concept, borrowed from conformal prediction literature [114], aims to make the test loss to "look exchangeable" with the source losses, thus adjusting for label shift. The main challenge here lies in reasonably approximating the likelihood ratio $P^t(Y)/P^s(Y)$, especially when facing simultaneous covariate and label shifts at test time. The ideas presented in [115] may offer a promising starting point for exploring this avenue.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We claimed to have contributed the following contributions:

   - We develop a sequential test for classification entropy drift detection: see Section 3.3, Appendix B, Appendix C, and Appendix E.
   - We show how to utilize the test martingale to analytically design a mapping function that transports the classifier entropies obtained at test time to resemble those of the source domain. The derivation of the algorithm is given in Sections 3.4, 3.5, and Appendix E.
   - We also conducted a wide range of experiments to support our approach in Sections 3.2, 4, and Appendix F.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We describe the problem setup and our assumptions in Section 2.1 and further discuss the limitations of our work in Section 5.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: The theoretical claims provided in Section 3.3, Appendix B, and Section 3.4 are proved in Appendix C.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: In Appendix F we provide all of the implementation details. The code used to conduct the experiments is attached to the submission as supplementary material. An open-source GitHub repository would be published upon publication.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: ImageNet and ImageNetC are both publicly available datasets. The data used in the synthetic experiment can be reproduced by running the code provided or implementing it based on Appendix F. All the results reported in the paper can be reproduced by running our software package.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: The experiments settings are described in Section 3.2 and Section 4. All the implementation details are extensively discussed in Appendix F.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We show the standard error in all graphs; we also report the standard errors in Tables 2 and 3.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: In Appendix F.2 we detail the computational resources we used to conduct the experiments in this work.

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The work is done in appropriation with `https://neurips.cc/public/EthicsGuidelines`.

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discussed the broader impacts of this work in Section 5.

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: We do not release anything that warrants a safeguard.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We credit prior contributions that we build upon, as well as provide links to the relevant code repositories. The open-source ImageNet and ImageNet-C datasets are also credited.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: While we do not provide new assets, we support the paper with a properly documented software package.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: No human subjects were used in this work.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: No human subjects were used in this work.