The Over-Certainty Phenomenon in Modern Test-Time Adaptation Algorithms

Anonymous authors
Paper under double-blind review

Abstract

When neural networks are confronted with unfamiliar data that deviate from their training set, this signifies a domain shift. While these networks output predictions on their inputs, they typically fail to account for their level of familiarity with these novel observations. Prevailing works navigate test-time adaptation with the goal of curtailing model entropy, yet they unintentionally produce models that struggle with sub-optimal calibration—a dilemma we term the over-certainty phenomenon. This over-certainty in predictions can be particularly dangerous in the setting of domain shifts, as it may lead to misplaced trust. In this paper, we propose a solution that not only maintains accuracy but also addresses calibration by mitigating the over-certainty phenomenon. Our method achieves state-of-the-art performance in terms of expected calibration error and negative log likelihood, all while maintaining parity in accuracy.

1 Introduction

When encountering new environments, humans naturally adopt a cautious approach, assimilating the novelty to guide their decision-making. This inherent ability to assess unfamiliarity and adjust certainty has not been entirely emulated in artificial neural networks. Unlike humans who might exhibit hesitation in unknown situations, many test-time adaptation (TTA) algorithms lack an explicit mechanism to modulate certainty in response to the novelty or unfamiliarity of their inputs.

Deep learning has never been a stranger to the challenges of uncertainty. Over the past few years, the miscalibration problem of modern neural networks has gained substantial attention, as highlighted by works such as Guo et al. (2017), Abdar et al. (2021), Liang et al. (2017), and Pampari & Ermon (2020). However, there is an observed void in the discussion of how TTA algorithms themselves alter calibration. In this paper, we uncover the *over-certainty phenomenon*, a phenomenon that plagues many modern TTA algorithms by harming model calibration.

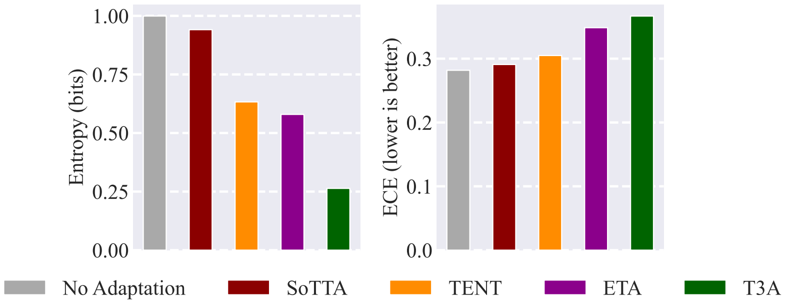

Figure 1: This chart shows four modern TTA algorithms adapting EfficientNet to the *clipart* domain. Minimizing entropy is a common objective in recent work. However, this can have consequences on model calibration.

A prevailing strategy among TTA algorithms is the minimization of entropy, either as an explicit target or as an inherent by-product of their methodology. And while this might bolster accuracy metrics, our research indicates a concerning trend: excessive entropy reduction can be detrimental to model calibration. What makes this trend more problematic is that it occurs within the context of a new domain, where epistemic uncertainty should typically be greater. **In our work, we measure certainty as the inverse of Shannon entropy** of a model's output after a softmax operation: $H(f(x))^{-1} = (\texttt{Entropy}_2(\texttt{SoftMax}(f(x))))^{-1}$.

To further frame our discussion, TTA is used when a model, trained on a source domain (X_s, Y_s), is presented with the challenges of a different yet analogous target domain (X_t) with no labels. **In our problem setting, we do not assume access to X_t until we adapt.** The nuances between these domains, commonly termed as domain shift, can introduce significant disruptions in model performance. TTA, in its essence, aspires to adapt the insights harvested from the source domain and apply them proficiently to the target domain, bypassing the need for labeled data in the latter. Thus, within the context of this domain shift, it can be especially problematic to be too certain.

With the purpose of addressing these intertwined challenges, we introduce *Dynamic Entropy Control.* This TTA technique seeks to augment accuracy and improve model calibration. By interweaving calibration into the core learning process, we produce a TTA adjustment algorithm that jointly improves accuracy while managing epistemic uncertainty. To summarize, our contributions are,

- The identification of the over-certainty phenomenon. We emphasize that our work is not about the well-studied tendency of models to become less calibrated under distribution shift Pampari & Ermon (2020); Ovadia et al. (2019). Instead, we show how a recent trend in the design of TTA algorithms can yield models that are even more miscalibrated than the unadapted baseline. To the best of our knowledge, no prior work—including Press et al. (2024)—explicitly analyzes this pattern while also providing thorough empirical evidence and mechanistic insight into why over-certainty emerges during adaptation.
- *Dynamic Entropy Control*, a new TTA algorithm that achieves SOTA calibration in all of the four datasets and SOTA accuracy uplifts in the majority of domain shifts.

Our study is scoped to test-time adaptation algorithms that follow the prevailing trend of minimizing entropy on unlabeled observations in the classification setting. While we acknowledge the existence of TTA methods that do not follow this paradigm (e.g., Liang et al. (2020); Tang et al. (2020)), they fall outside the focus of our analysis. **Unless otherwise stated, all mentions of "TTA algorithms" in this work refer specifically to entropy-reduction–based approaches.**

1.1 Why Care About Over-Certainty?

We emphasize that calibrating uncertainty estimates has ramifications that go well beyond quantitative metrics. For instance, in real-world scenarios that rely on trustworthy estimates of model confidence (e.g., autonomous driving, active learning, anomaly detection), an overconfident model can lead to harmful misjudgments You et al. (2022). As a concrete illustration, consider a self-driving car that uses a classifier to detect whether a neighboring lane is occupied:

- *Pre-adaptation (low certainty):* The vehicle's classifier might cautiously predict safe to "switch lanes" but at a low confidence. This low certainty prevents the car from making a risky maneuver.
- *Post-adaptation (artificially high certainty):* After applying a traditional TTA algorithm, the same classifier might become *overly certain* about its safe to switch lanes" prediction. In reality, if this confidence is misplaced, the system can make a high-stakes error that jeopardizes safety.

Thus, our approach seeks to preserve or improve classification accuracy *and* produce calibrated uncertainty estimates, ensuring that decisions made based on confidence thresholds are more reliable. Such reliability is also valuable in other domains that hinge on calibrated outputs, such as fairness-sensitive applications

Hébert-Johnson et al. (2018); Creager et al. (2021), where miscalibration can exacerbate biases or distort outcome distributions.

Fairness Considerations. As suggested in Pleiss et al. (2017), there is a growing recognition of how calibration can influence fairness. Overconfident models may disadvantage certain groups when high-confidence predictions guide resource allocation or risk assessments. Ensuring well-calibrated probabilities is therefore crucial but often overlooked in current TTA approaches.

2 Related Work

Our literature survey covers methodologies catered towards updating a neural network on unlabeled data. For the sake of brevity, we will refer to unlabeled data as "observations." This section gives an overview of the work done to improve networks on the fly. We start by introducing earlier work, such as dictionary learning techniques and lead our way into recent developments. We elaborate the algorithms we compare with in detail to give mechanistic insight on why over-certainty happens. The next section covers how calibration is measured and improved in the context of a domain shift. Lastly, we discuss assessing the reliability of an observation. Furthermore, we provide a supplementary discussion of related work in Appendix A.3.

2.1 The Test-Time Entropy Reduction Paradigm

The phrase "self-taught learning" was coined by Raina et al. (2007). In this work, the authors utilize observations to find an optimal sparse representation of said observations. This sparse representation is used to train their model in lieu of the ordinary training set to improve out-of-distribution (OOD) performance.

Work in this field has extended to a variety of approaches such as the use of pseudo-labeling to exploit the existing model's predictions as target labels Lee et al. (2013); Mancini et al. (2018); Wang et al. (2022). Pseudo-labeling can be thought of under the guise of knowledge distillation (KD) Hinton et al. (2015); Gou et al. (2021). KD is a transfer learning paradigm where a large neural network, known as the teacher, transfers "knowledge" to a smaller "student" network. Succinctly, the student is trained to match the output of the teacher when given the same input as the teacher Stanton et al. (2021). In pseudo-labeling, the teacher and student are the same network.

The TENT algorithm introduces the trend of **test-time entropy minimization** Wang et al. (2020a). Entropy minimization can be thought of as using pseudo labels with the cross entropy loss function. In other words, the entropy minimization works by using gradient descent to minimize:

$$L_{TENT} = -\sum_{y \in C} f(y|x) \log f(y|x) \tag{1}$$

to update the model's batch-normalization parameters.

More recent advancements in this trend include EATA/ETA Niu et al. (2022), T3A Iwasawa & Matsuo (2021) and SoTTA Gong et al. (2024). ETA[1] advances on TENT by making sure that observations are *reliable* and *non-redundant* before they are used for updating the batch-normalization parameters. To do this, they compute a sample adaptive weight, $\mathcal{S}(x)$, for each observation before minimizing entropy:

$$L_{ETA} = -\mathcal{S}(x) \sum_{y \in C} f(y|x) \log f(y|x) \tag{2}$$

where $\mathcal{S}(x)$ is a function of the entropy of the model towards the batch sample (i.e., the reliability) and the similarity to what it has seen before (i.e., non-redundancy). Similar to the aforementioned methods, SoTTA minimizes entropy via sharpness-aware-minimization (SAM) Foret et al. (2020). The algorithm employs high-confidence uniform sampling to create a memory bank of size N_{SoTTA} which stores reliable and class-balanced observations. This is done by using confidence to asses if an observation should be used for

---

[1] While the authors of EATA/ETA introduce two similar algorithms, for our paper, we focus on ETA which performs the best between the two.

adaptation. Confidence is defined as:

$$\mathcal{C}_f(x) = \max_{i=1,\ldots,n} \frac{\exp(f(x)_i)}{\sum_{j=1}^{n} \exp(f(x)_j)} \tag{3}$$

If $\mathcal{C}_f(x) > \mathcal{C}_0$, where $\mathcal{C}_0$ is some pre-defined confidence threshold, then x is saved into memory. Afterwards, the SAM optimizer is used to minimize entropy (equation 1) via two backpropagation steps. The T3A algorithm Iwasawa & Matsuo (2021) differs from the previous three as it focuses on updating the *prototypes* Snell et al. (2017) of each class during test time:

$$S_k^t = \begin{cases} S_k^{t-1} \cup \{f(x)\}, & \text{if } \hat{y} = y_k \\ S_k^{t-1}, & \text{else.} \end{cases} \tag{4}$$

$$c_k = \frac{1}{|S_k|} \sum_{z \in S_k} z \tag{5}$$

where c_k represents the centroid of the prototypes of a class $k \in C$, where C is the number of classes. We define the feature extractor, ψ, as all the layers of the backbone before the final dense layer. The final dense layer, ϕ, is what we refer to as the classifier, it is composed of the class centroids. We denote the output of the feature extractor as $z = \psi(x)$.

Unlike TENT, SoTTA, or ETA, T3A does not explicitly reduce entropy as it does not use a loss function, however the authors claim that entropy reduction is an effect of using their algorithm. Similar to ETA, this algorithm filters less reliable samples during equation 4 by only keeping the M lowest entropy prototypes for each class. Therefore, the algorithm stores $C \cdot M$ prototypes.

2.2 Neural Network Calibration

Neural network calibration has been of intense interest in recent years due to the critical role of confidence values, which reflect the probability assigned to predictions, in various applications. For instance, BranchyNet Teerapittayanon et al. (2017) uses neural network confidence to enable early exits for faster inference, relying on high confidence at intermediate layers. However, Zhu et al. (2022) highlights the prevalent issue of certainty calibration in deep networks, where models often display overconfidence or underconfidence, likely due to overfitting during training. The authors of Ovadia et al. (2019); Minderer et al. (2021) explore this concept further. They measure a model's expected calibration error (ECE) and Negative Log-Likelihood (NLL). ECE measures how closely the confidence levels of a model's predictions match the actual probability of those predictions being correct. It calculates the average absolute difference between predicted confidence and the true outcome frequencies, providing a metric for the reliability of the model's probabilistic outputs. NLL, on the other hand, captures both the model's calibration and sharpness by quantifying how well the predicted probabilities align with the observed outcomes, penalizing overconfident yet incorrect predictions more heavily. **We follow their convention and use ECE and NLL as our measures for calibration error.** They notice models calibrated on the validation set tend to be well calibrated on the test set, but are not properly calibrated to shifted data.

Recent work has also investigated solutions to this phenomenon. Guo et al. (2017) discusses a technique known as temperature scaling while Wei et al. (2022) approaches this problem by regularizing the logit norm. More classical solutions to this problem exist as well; Zhang et al. (2021) and Müller et al. (2019) consider label smoothing to address this issue. Note that these techniques are addressed at calibrating the underlying backbone but have *not* been investigated with respect to TTA algorithms themselves.

2.3 Detecting Out of Distribution Data and Assessing Reliability

An increasing body of research examines how to assess whether and how closely a new observation aligns with a model's training distribution. For example, the authors of Tian et al. (2019) observe that if an autoencoder was trained to reconstruct inliers, it would have a greater reconstruction error when reconstructing OOD data. Schlegl et al. (2017) and Zenati et al. (2018) approach this issue by observing that the discriminator of

Table 1: Comparison of Shannon Entropy, ECE, and NLL across the Home Office dataset domains. Results use the MobileNet backbone. **Our reduction in ECE and NLL is statistically significant** ($p < 0.01$) while maintaining competitive accuracy.

Algorithm	**Shannon Entropy**				**ECE (lower is better)**				**NLL (lower is better)**			
	Art	**Clipart**	**Product**	**Real World**	**Art**	**Clipart**	**Product**	**Real World**	**Art**	**Clipart**	**Product**	**Real World**
No Adapt	0.950	0.951	0.704	0.689	0.302	0.316	0.183	0.169	3.196	3.330	1.777	1.638
DEC (ours)	2.239	2.615	2.081	1.795	**0.080**	**0.047**	**0.039**	**0.018**	**2.077**	**2.127**	**1.328**	**1.212**
T3A	0.188	0.231	0.138	0.170	0.439	0.418	0.236	0.248	7.904	7.756	3.536	3.321
ETA	0.940	0.966	0.735	0.678	0.313	0.324	0.208	0.172	3.180	3.438	1.966	1.768
TENT	0.918	0.895	0.653	0.636	0.299	0.308	0.187	0.172	3.120	3.234	1.780	1.632
SoTTA	0.977	0.971	0.701	0.685	0.279	0.301	0.186	0.167	2.938	3.240	1.825	1.632

a GAN learns whether or not a given input is an inlier. Many other works delve into this domain Bendale & Boult (2016); Ming et al. (2022); Park et al. (2023); Jiang et al. (2023); Wu et al. (2023); Miao et al. (2023); Fang et al. (2022). Regarding the TTA algorithms we compare against, the most common proxy for reliability is entropy on the observation.

3 Proposed Approach

3.1 The Over-Certainty Phenomenon

In this work, we present evidence for what we dub the *over-certainty phenomenon* (OCP) of contemporary TTA approaches. This phenomenon is that TTA algorithms tend to miscalibrate their underlying backbone networks by causing their predictions to be excessively certain. Modern TTA algorithms often strive to decrease test-time entropy. However, as shown in Fig. 1, this entropy reduction may increase ECE and NLL because the models become overly certain on their predictions.

This phenomenon of existing algorithms causing models to become overly certain presents itself across many other datasets. A compelling example is given in Table 2 which agglomerates calibration errors over 15 domain shifts; in this table, we see a clear trend of entropy reduction (certainty increasing) and sub-optimal calibration. Another example is provided in Table 1, T3A reduces entropy by a factor of about 4 in the *art, clipart* and *product* domains. As before, it causes ECE to worsen compared to the baseline. We do not claim that TTA algorithms should *always* strive to increase backbone uncertainty; poor calibration can also be caused by under-certainty and there exist cases where reducing entropy compared to baseline improves calibration. However, we find that the resulting calibration is still sub-optimal. Despite these complexities, our investigation reveals a consistent pattern: *the over-certainty phenomenon causes sub-optimal model calibration*, a significant concern for safety, robustness, and reliability.

3.2 What Causes the Over-Certainty Phenomenon?

We identify two plausible causes of the OCP, the first issue is that modern TTA algorithms aim at minimizing backbone entropy too aggressively. In the case of TENT, ETA, and SoTTA, their loss functions, equation 1 and equation 2, explicitly aim at reducing a model's entropy. Regarding TENT, there is no regularization of this process. In the case of ETA, the algorithm uses a *reliability score*, $S(x)$, which aims at weighing observations differently but does not regularize the distributions of the pseudo-labels. Unlike the other two, SoTTA minimizes entropy twice per iteration. The authors of T3A claim that entropy reduction is an effect of using their algorithm. In fact, they show in certain datasets T3A reduces entropy more than TENT does.

Another issue is how existing methods evaluate observation *reliability*, the suitability of a model's prediction for use for adaptation. Previous works, ETA and T3A, tap into the power of model certainty, using it to weigh the influence of observations. ETA assesses reliability by ensuring that observations meet a certain entropy threshold; similarly, T3A uses entropy to sort the importance of class prototypes. However, as prior work has shown, there are drawbacks in using entropy as a proxy for reliability in this manner Wei et al. (2022). To illustrate our point, we give a toy example of how using entropy can lead to a misleading conclusion:

Example 1. *Suppose that we analyze the classifier while classifying between two classes with class centroids, c_0 and c_1. This is done by taking the output of the feature extractor, $\psi(x) = z$, and computing the dot product between the centroids and z.*

$$g = [z \cdot c_0, z \cdot c_1] \tag{6}$$

Consider g_{t1}, g_{t2} and g_s as vectors representing the inner products related to two observations, x_{t1} and x_{t2}, and to a specific training sample, x_s. Specifically, g corresponds to the dot products between the output of the feature extractor and the class centroids. As an example, let's assume:

$$g_s = [8.0, 7.29]; \;\; g_{t1} = [1.92, 1.00]; \;\; g_{t2} = [6.10, 6.50];$$

If we take the softmax of these vectors and compute the entropy, we get $\texttt{Entropy}_2(\texttt{SoftMax}(g))$ *for g_s, g_{t1} and g_{t2}, as 0.92 bits, 0.86 bits and 0.97 bits, respectively.*

Notice that if we consider the entropy of these three vectors as a proxy for reliability, we would consider x_{t1} to be more reliable than x_{t2}, despite x_{t2} having considerably greater inner product with the class centroids. It is highly likely that the values of g_{t1} occurred due to spurious feature correlations between x_{t1} and the class centroids. In fact, in the scenario above, x_{t1} would be deemed to be more reliable than the genuine source domain observation x_s. Note that an analogous remark could be made on using confidence instead of entropy. Our analysis is not contrived; in Fig. 2, as we increase the domain shift intensity, the observation logit norm decays and has higher variance.

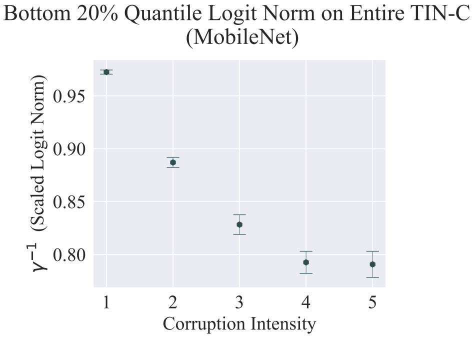

Figure 2: As domain shift intensity increases, the bottom quantile of the observation logit norm decreases. The γ^{-1} value (step 8) from our `CCR` algorithm represents the ratio between the l_2 norms of the observation logits, z, and the training-set logits, κ. As z decreases, γ increases; this regularizes low-logit-norm observations more aggressively (step 9). Vertical bars indicate domain-to-domain standard deviations.

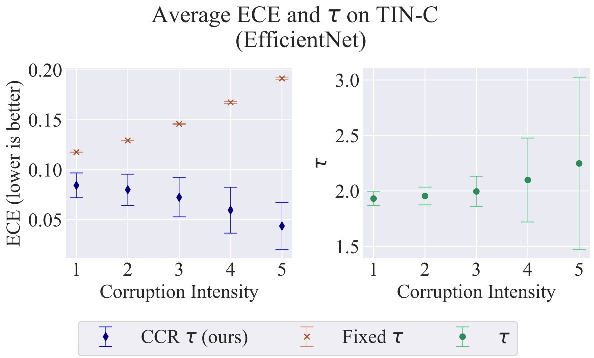

Figure 3: Our ablation study highlights the effectiveness of the `CCR` algorithm compared to a fixed τ optimized for minimal ECE on the source-domain training set. The rightmost figure displays the computed τ values, with vertical bars indicating domain-to-domain standard deviations.

Furthermore, existing approaches do not consider a model's certainty on the source domain. For example, ETA's reliability paradigm rejects observations which have $H \geq 0.4 \cdot \ln(C)$. What if we *expect* (i.e. there was high entropy on the training set) the model to have high entropy? By only evaluating the target domain's certainty without juxtaposing it against the source domain certainty, there is a lack of reference in terms of assessing the reliability of the observation. We address this issue via the h_0 parameter explained in the next section.

3.3 The Dynamic Entropy Control TTA Algorithm

To ameliorate the over-certainty phenomenon, we introduce Dynamic Entropy Control (`DEC`) (Algorithm 2). DEC refines the model's certainty levels, aligning them more closely with its actual accuracy, by selectively adjusting the temperature parameter during the pseudo-labeling process (line 6). This is achieved without directly altering ground truth labels, instead focusing on the tempering of logits through temperature adjustments. **Our approach can be implemented using a single model which alternates weights. To facilitate understanding, we explain our algorithm as if it uses two distinct models**: the teacher and the student. The "teacher" model is simply f before any adaptation. The "student" model is simply a copy of f which we use to store the weight adaptations.

Algorithm 1 `Compute Certainty Regularizer (CCR)`

```
Input: $f_{te}(X), h_0, h_{max}, t_{min}, t_{max}, \kappa$
Output: $T_{vec}$
 1: $Z = f_{te}(X)$ {get logits}
 2: $H_{vec} = \texttt{Entropy}_2(\texttt{SoftMax}(Z))$ {entropy for each sample}
 3: $H_{diff} = H_{vec} - h_0$ {compare entropy with source entropy}
 4: $H_{scaled} = \texttt{sigmoid}(H_{diff}/\sqrt{h_{max}})$ {scale between [0,1]}
 5: Init. $T_{vec}$
 6: for $h_i \in H_{scaled}$ and $z_i \in Z$ do
 7:     $t_i = t_{min} + h_i \cdot (t_{max} - t_{min})$
 8:     $\gamma_i = \frac{\kappa}{\|z_i\|_2}$ {Scale the logit norm}
 9:     $\tau_i = \gamma_i \cdot t_i$ {Adjust regularizer via logit norm}
10:     Store $T_{vec} \leftarrow \tau_i$
11: end for
12: return $T_{vec}$
```

Algorithm 2 `Dynamic Entropy Control (DEC)`

```
Input: $f_{te}, f_s, h_0, X, \kappa$
Parameters: $t_{min}, t_{max}, h_{max}, \lambda$
Output: $f_s^+$
 1: $T_{vec} = \texttt{CCR}(f_{te}(X), h_0, \kappa, h_{max}, t_{min}, t_{max})$
 2: Init $loss$
 3: for $x_i \in X$ and $\tau_i \in T_{vec}$ do
 4:     $s_{si} = \texttt{SoftMax}_{T=1}(f_s(x_i))$
 5:     $s_{ti} = \texttt{SoftMax}_{T=\tau_i}(f_{te}(x_i))$ {smoothen teacher labels}
 6:     $l = (\texttt{avg}(T_{vec}))^2 \cdot \texttt{BCE}(s_{si}, s_{ti})$
 7:     Store $loss \leftarrow l$
 8: end for
 9: $L = \texttt{avg}(loss)$
10: $f_s' \leftarrow \theta_s - \lambda \nabla L(\theta_s)$
11: $f_s^+ = \texttt{Temperature\_Scale}(f_s', \texttt{avg}(T_{vec}))$
12: return $f_s^+$
```

This process involves adjustments of the student model, guided by the comparative analysis of entropy and logit norms, thereby fostering a more accurate and reliable predictive model. The inputs f_{te}, f_s, h_0, and X correspond with the teacher model, the student model, the teacher's average entropy on the training set, and unlabeled observations, respectively. The input κ is the median l_2 norm of the training-set logits; this gives us a context in terms of logit norms. The λ parameter is the learning rate for SGD, which we set to 0.001 for all experiments.

`Compute Certainty Regularizer` (Algorithm 1) returns a regularizer, $\tau_i \in T_{vec}$, for each observation, with respect to relative observation entropy and relative logit-norm. The parameter $h_0 = \mathbb{E}[\texttt{Entropy}_2(\texttt{SoftMax}(f_{te}(X_s)))]$ is used as a reference point. Intuitively, the idea is to compare the model's certainty on the observation with respect to the certainty of what it was trained on. Parameters t_{min} and t_{max} are used to shift the entropies into a valid range[2]. $h_{max} = \log_2(C)$ (maximum entropy is when all classes are equiprobable) is used to smoothen the sigmoidal function.

Parameter τ returned by Algorithm 1 plays a pivotal role. We name it the *certainty regularizer*. It regulates the "sharpness" of predicted probabilities and smoothens the pseudo labels produced by the teacher. As τ increases, the certainty of the prediction decreases. In other words, τ should be greater for less reliable observations.

By preventing the model from becoming inappropriately certain in its predictions, we produce a model that is better calibrated—its prediction certainty more closely aligns with its true accuracy. In `DEC`, we do not label smooth directly, but instead adjust the temperature parameter of our teacher during the adaptation process. To show how `DEC` regularizes observations appropriately, we continue from Example 1 to Example 2:

Example 2. *Given the same g_{t1}, g_{t2} and g_s from Example 1, we input these into our `CCR` algorithm. We set $h_0 = \texttt{Entropy}_2(\texttt{SoftMax}(g_s)), \kappa = |g_s|_2, t_{min} = 1.0$ and $t_{max} = 2.0$. Our algorithm first computes a*

---

[2] Because τ guides the temperature scaling component of our algorithm, we follow the convention of Guo et al. (2017) and recommend setting $t_{min/max}$ within $[1.0, 3.0]$.

scaled entropy, H_{scaled} with respect to the source domain entropy for g_s, g_{t1} and g_{t2}, as 0.49, 0.48, and 0.50, respectively.

In step 7 of `CCR`*, this entropy is transformed into a preliminary regularizer, t_i. Then, step 9 adjusts t_i by considering logit norm with respect to the source-domain logit norm.*

$$\tau_{g_s} = 1.49; \quad \tau_{g_{t2}} = 7.42; \quad \tau_{g_{t1}} = 1.83$$

Notice that, unlike purely entropy-based methods, the `CCR` algorithm correctly assigns greater regularization to the less reliable samples. Namely, step 9 ensures that samples that are low-entropy due to degenerate reasons are properly regularized by considering logit norm. Furthermore, unique from existing algorithms, our regularizer directly addresses model certainty. The impact of `CCR` is analyzed in Fig. 3.

An interpretation as to how our model improves accuracy is through the works of Xie et al. (2020) and Lukasik et al. (2020). Although the former's work concerns itself in the semi-supervised learning setting, we found their observations to be relevant. That is, they introduce the *noisy student,* a network that has been *noised* by dropout and stochastic-depth. They find that their noisy student can even learn to outperform the teacher which initially produced the pseudo-labels. For the latter work, they establish that label smoothing mitigates label noise, which is a desirable property with respect to unsupervised adaptation. Specifically, they find that label smoothing can be thought of as a regularizer. This motivates us to smooth more aggressively when we notice that an observation might be less reliable.

4 Experiments

4.1 Experimental Setting

In order to evaluate DEC, we conduct a series of experiments using three different backbone models across four datasets. Our primary evaluation metrics will be model accuracy, NLL and $\text{ECE}_{bins=15}$ on the observations, allowing us to examine both the predictive performance and the calibration quality of the models. By using varied domains and different backbone architectures, we aim to demonstrate the robustness and adaptability of our algorithm in handling diverse and challenging TTA scenarios. All experiments are run three times. Note that the $t_{min/max}$ hyperparameters are set per dataset; we do not set these per domain shift. We present accuracy and calibration for all datasets along with experiment variances either in the main paper or Appendix A.4 & A.5.

We compare with TENT, T3A, SoTTA and ETA, four recent TTA algorithms which follow the test-time entropy reduction paradigm. We do a single iteration of adaptation for all algorithms unless stated otherwise. Dataset preprocessing steps and **a discussion of hyperparameters for all algorithms are in more detail in the Appendix A.2.**

4.2 Datasets

The following publicly available TTA datasets are used in our experiments; we selected these because they are commonly used in existing works and provide a variety of domain shifts. In total, we evaluate our algorithm over **26 domain shifts which include a total of 282 classes.** Furthermore, 15 of our domain shifts have 5 corruption levels. For some datasets, we tested using the "leave one out" (LOO) paradigm; for example, in PACS, to test generalization to *pictures*, we first trained our backbone networks on *art, cartoon, sketch* before adapting.

1. PACS Li et al. (2017) has 4 domains: *pictures, art, cartoon, sketch* with 7 classes. Tested using LOO.
2. HomeOffice Venkateswara et al. (2017) has 4 domains: *art, clipart, product, real* with 65 classes. Tested using LOO.
3. Digits is a combination of 3 "numbers" datasets: USPS Hull (1994), MNIST LeCun et al. (2010), and SVHN Netzer et al. (2011). There are 10 classes. Tested using LOO by training on the source domains' training sets and adapting to target domain's test set.

4. TinyImageNet-C (TIN-C) Le & Yang (2015), has 15 domains with 200 classes. Backbones are trained on corruption-free (source) training set, adapted to and evaluated on corrupted (target) domains. For each target domain, there are 5 tiers of corruption.

4.3 Back Bones and Training Details

We test all but the Digits dataset on two popular classifiers, EfficientNetB0 Tan & Le (2019) and MobileNet Howard et al. (2017) pre-trained for ImageNet Deng et al. (2009). We flatten the output of both networks and add a final dense layer with an output shape equivalent to the number of classes.

We evaluate the Digits dataset using "SmallCNN", which is a minor adaptation of the original LeNet LeCun et al. (1998) with batch normalization layers and max pooling. This serves to represent more compact and straightforward architectures for less complex datasets. The specific details and orderings of the layers in SmallCNN are elaborated on in Appendix A.6. Note that all three models use batch normalization layers as necessitated by ETA, TENT, and SoTTA.

5 Results

We present our accuracy, ECE and NLL measurements on the four aforementioned datasets. To show evidence of the over-certainty phenomenon, we also report prediction entropy. More comprehensive figures/tables can be found in the appendix. To show the impact of our `CCR` algorithm, which produces our certainty regularizer τ, we perform an ablation experiment in Fig. 3.

Table 2: Comparison of Shannon entropy, ECE, and NLL averaged across the 15 domain shifts of TIN-C. We use the EfficientNet backbone. Standard deviations across domains are shown as subscripts. This experiment highlights how excessive certainty (low Shannon entropy) correlates with sub-optimal calibration. **Our reduction in ECE and NLL is statistically significant ($p < 0.01$)** while maintaining competitive accuracy (see Tables 6 and 8).

Algorithm	**Shannon Entropy**					**ECE (lower is better)**					**NLL (lower is better)**				
	Tier 1	**Tier 2**	**Tier 3**	**Tier 4**	**Tier 5**	**Tier 1**	**Tier 2**	**Tier 3**	**Tier 4**	**Tier 5**	**Tier 1**	**Tier 2**	**Tier 3**	**Tier 4**	**Tier 5**
No Adapt	$2.00_{0.17}$	$2.10_{0.22}$	$2.25_{0.30}$	$2.44_{0.51}$	$2.63_{0.73}$	$0.23_{0.02}$	$0.26_{0.03}$	$0.29_{0.03}$	$0.32_{0.04}$	$0.34_{0.04}$	$3.06_{0.31}$	$3.40_{0.45}$	$3.92_{0.70}$	$4.57_{0.95}$	$5.15_{1.07}$
DEC (ours)	$4.48_{0.25}$	$4.68_{0.34}$	$4.90_{0.53}$	$5.01_{0.77}$	$5.32_{0.80}$	$\mathbf{0.08}_{0.01}$	$\mathbf{0.08}_{0.02}$	$\mathbf{0.07}_{0.02}$	$\mathbf{0.06}_{0.02}$	$\mathbf{0.04}_{0.03}$	$\mathbf{2.27}_{0.12}$	$\mathbf{2.44}_{0.20}$	$\mathbf{2.69}_{0.35}$	$\mathbf{3.04}_{0.59}$	$\mathbf{3.38}_{0.68}$
T3A	$1.36_{0.11}$	$1.41_{0.14}$	$1.46_{0.17}$	$1.60_{0.20}$	$1.74_{0.23}$	$0.33_{0.03}$	$0.35_{0.03}$	$0.38_{0.03}$	$0.41_{0.04}$	$0.43_{0.04}$	$3.63_{0.39}$	$4.09_{0.72}$	$5.10_{1.80}$	$7.13_{4.41}$	$9.82_{7.09}$
ETA	$1.37_{0.08}$	$1.47_{0.14}$	$1.57_{0.23}$	$1.71_{0.38}$	$1.87_{0.67}$	$0.37_{0.03}$	$0.40_{0.03}$	$0.41_{0.04}$	$0.43_{0.04}$	$0.44_{0.04}$	$4.45_{0.52}$	$4.93_{0.72}$	$4.89_{0.64}$	$5.63_{0.79}$	$6.29_{1.20}$
TENT	$1.40_{0.08}$	$1.48_{0.12}$	$1.57_{0.19}$	$1.62_{0.33}$	$1.79_{0.49}$	$0.27_{0.02}$	$0.30_{0.03}$	$0.33_{0.03}$	$0.36_{0.04}$	$0.39_{0.04}$	$3.01_{0.22}$	$3.36_{0.41}$	$3.89_{0.75}$	$4.54_{1.14}$	$5.20_{1.33}$
SoTTA	$1.71_{0.07}$	$1.78_{0.12}$	$1.88_{0.18}$	$2.08_{0.31}$	$2.26_{0.41}$	$0.20_{0.02}$	$0.21_{0.02}$	$0.23_{0.03}$	$0.25_{0.03}$	$0.28_{0.03}$	$2.56_{0.13}$	$2.75_{0.23}$	$3.04_{0.40}$	$3.49_{0.75}$	$3.98_{1.04}$

5.1 DEC Reduces Calibration Error

Due to our algorithm addressing the over-certainty phenomenon, we significantly improve calibration performance. DEC achieves state-of-the-art average ECE and NLL in all tested datasets and in nearly all individual domain shifts. We recognize that reducing entropy *did* improve calibration compared to baseline in some cases, but the resulting calibration was still sub-optimal. Fig. 3 empirically validates our finding that an adaptive certainty regularizer aids in reducing ECE and NLL. Moreover, the variance of τ increases as the corruption intensity increases; indicating a broader dynamic range of regularization when encountering more difficult observations.

5.2 DEC Augments Accuracy

In addition to strong calibration performance, DEC provides consistent accuracy uplifts while not necessitating any transformations on observations. By jointly exploiting backbone entropy and logit norm (see Fig. 2), we are able to effectively assess observation reliability. After doing so, we apply greater regularization to less reliable observations. This, in turn, allows us to mitigate the potential label noise produced by the pseudo-labels.

Tables 6, 4 and 8 show that our algorithm maintains competitive accuracy with recent test-time entropy minimization approaches while addressing the over-certainty phenomenon. Moreover, unlike SoTTA and T3A, we do not store observations or training samples during the adaptation process as this could potentially cause security or privacy issues during deployment.

Table 3: Average accuracy, ECE, entropy, and NLL on Digits and PACS datasets tested with LOO. Our approach achieves the best average ECE and NLL. Domain-to-domain σ^2_{max} values for accuracy, ECE, entropy, and NLL are reported.

Algorithm	**Accuracy**	**ECE**	**Entropy**	**NLL**
No Adapt	0.589	0.301	0.439	4.465
DEC (ours)	**0.648**	**0.169**	1.220	**1.422**
T3A	0.625	0.271	1.873	1.889
ETA	0.646	0.270	0.356	3.779
TENT	0.645	0.262	0.398	3.252
SoTTA	0.640	0.252	0.474	3.222
σ^2_{max}	0.220	0.180	0.200	46.240

Table 4: Performance on the Digits dataset using the SmallCNN backbone.

Algorithm	**Accuracy**	**ECE**	**Entropy**	**NLL**
No Adapt	0.873	0.105	0.081	1.182
DEC (ours)	0.879	**0.061**	0.233	**0.466**
T3A	**0.897**	0.084	0.065	0.866
ETA	0.878	0.102	0.071	1.213
TENT	0.880	0.101	0.072	1.160
SoTTA	0.887	0.096	0.062	1.188
σ^2_{max}	0.020	0.014	0.075	1.848

Table 5: Performance on the PACS dataset using the EfficientNet backbone.

Table 6: Accuracy on TIN-C with MobileNet backbone across different tiers of corruption. Standard deviations across the domain shifts are shown as subscripts.

Algorithm	**Tier 1**	**Tier 2**	**Tier 3**	**Tier 4**	**Tier 5**
No Adapt	$0.27_{0.04}$	$0.23_{0.05}$	$0.19_{0.07}$	$0.15_{0.08}$	$0.12_{0.08}$
DEC (ours)	$\mathbf{0.39}_{0.02}$	$\mathbf{0.37}_{0.03}$	$\mathbf{0.33}_{0.05}$	$\mathbf{0.29}_{0.07}$	$0.24_{0.08}$
T3A	$0.27_{0.04}$	$0.24_{0.05}$	$0.20_{0.07}$	$0.16_{0.08}$	$0.13_{0.08}$
ETA	$0.22_{0.05}$	$0.19_{0.06}$	$0.14_{0.07}$	$0.11_{0.07}$	$0.08_{0.06}$
TENT	$0.37_{0.03}$	$0.34_{0.04}$	$0.30_{0.06}$	$0.25_{0.08}$	$0.20_{0.09}$
SoTTA	$\mathbf{0.39}_{0.02}$	$\mathbf{0.37}_{0.03}$	$\mathbf{0.33}_{0.04}$	$\mathbf{0.29}_{0.06}$	$\mathbf{0.25}_{0.08}$

5.3 Discussion

We would like to underscore our algorithm's robustness to the choice of $t_{min/max}$. **We performed our experiments by selecting our hyperparameters per dataset, NOT per domain shift.** As shown in Tables 2 and 7, we achieve statistically significant reduction in ECE despite using the same hyperparameters across **15 domain shifts**. Our results in Table 1 further bolster our claims of robustness. Again, our hyperparameters are fixed across the four domain shifts but we still achieve statistically significant improvement.

Our study identifies the *over-certainty phenomenon* of modern TTA methodologies which cause harm to model calibration. To ameliorate this issue, we introduce a certainty regularizer, τ, that modulates model entropy and mitigates pseudo-label noise. The resulting algorithm, DEC, jointly improves model accuracy and reduces calibration error. DEC does not require batch normalization layers like SoTTA, TENT, and ETA do. This permits greater freedom when choosing a backbone. Furthermore, DEC is compatible with existing prototypical learning approaches. We will release our code after acceptance.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey

Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `https://www.tensorflow.org/`. Software available from tensorflow.org.

Moloud Abdar, Farhad Pourpanah, Sadiq Hussain, Dana Rezazadegan, Li Liu, Mohammad Ghavamzadeh, Paul Fieguth, Xiaochun Cao, Abbas Khosravi, U. Rajendra Acharya, Vladimir Makarenkov, and Saeid Nahavandi. A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information Fusion*, 76:243–297, 2021. ISSN 1566-2535. doi: https://doi.org/10.1016/j.inffus.2021.05.008.

Abhijit Bendale and Terrance E Boult. Towards open set deep networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1563–1572, 2016.

Elliot Creager, Jörn-Henrik Jacobsen, and Richard Zemel. Environment inference for invariant learning. In *International Conference on Machine Learning*, pp. 2189–2200. PMLR, 2021.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Zhen Fang, Yixuan Li, Jie Lu, Jiahua Dong, Bo Han, and Feng Liu. Is out-of-distribution detection learnable? *Advances in Neural Information Processing Systems*, 35:37199–37213, 2022.

Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. *arXiv preprint arXiv:2010.01412*, 2020.

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.

Yunye Gong, Xiao Lin, Yi Yao, Thomas G Dietterich, Ajay Divakaran, and Melinda Gervasio. Confidence calibration for domain generalization under covariate shift. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 8958–8967, 2021.

Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129:1789–1819, 2021.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International conference on machine learning*, pp. 1321–1330. PMLR, 2017.

Ursula Hébert-Johnson, Michael Kim, Omer Reingold, and Guy Rothblum. Multicalibration: Calibration for the (computationally-identifiable) masses. In *International Conference on Machine Learning*, pp. 1939–1948. PMLR, 2018.

Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*, 2015. URL `http://arxiv.org/abs/1503.02531`.

Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications, 2017. URL `https://arxiv.org/abs/1704.04861`.

J.J. Hull. A database for handwritten text recognition research. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(5):550–554, 1994. doi: 10.1109/34.291440.

Yusuke Iwasawa and Yutaka Matsuo. Test-time classifier adjustment module for model-agnostic domain generalization. *Advances in Neural Information Processing Systems*, 34:2427–2440, 2021.

Xue Jiang, Feng Liu, Zhen Fang, Hong Chen, Tongliang Liu, Feng Zheng, and Bo Han. Detecting out-of-distribution data through in-distribution class prior. In *International Conference on Machine Learning*, pp. 15067–15088. PMLR, 2023.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014. URL `https://arxiv.org/abs/1412.6980`.

Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

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, pp. 896. Atlanta, 2013.

Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M Hospedales. Deeper, broader and artier domain generalization. In *Proceedings of the IEEE international conference on computer vision*, pp. 5542–5550, 2017.

Jian Liang, Dapeng Hu, and Jiashi Feng. Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation. In *International conference on machine learning*, pp. 6028–6039. PMLR, 2020.

Shiyu Liang, Yixuan Li, and Rayadurgam Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks. *arXiv preprint arXiv:1706.02690*, 2017.

Michal Lukasik, Srinadh Bhojanapalli, Aditya Menon, and Sanjiv Kumar. Does label smoothing mitigate label noise? In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 6448–6458. PMLR, 13–18 Jul 2020. URL `https://proceedings.mlr.press/v119/lukasik20a.html`.

Massimiliano Mancini, Samuel Rota Bulo, Barbara Caputo, and Elisa Ricci. Best sources forward: domain generalization through source-specific nets. In *2018 25th IEEE international conference on image processing (ICIP)*, pp. 1353–1357. IEEE, 2018.

Wenjun Miao, Guansong Pang, Tianqi Li, Xiao Bai, and Jin Zheng. Out-of-distribution detection in long-tailed recognition with calibrated outlier class learning. *arXiv preprint arXiv:2312.10686*, 2023.

Matthias Minderer, Josip Djolonga, Rob Romijnders, Frances Hubis, Xiaohua Zhai, Neil Houlsby, Dustin Tran, and Mario Lucic. Revisiting the calibration of modern neural networks. *Advances in Neural Information Processing Systems*, 34:15682–15694, 2021.

Yifei Ming, Ziyang Cai, Jiuxiang Gu, Yiyou Sun, Wei Li, and Yixuan Li. Delving into out-of-distribution detection with vision-language representations. *Advances in Neural Information Processing Systems*, 35: 35087–35102, 2022.

Rafael Müller, Simon Kornblith, and Geoffrey E. Hinton. When does label smoothing help? *CoRR*, abs/1906.02629, 2019. URL `http://arxiv.org/abs/1906.02629`.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. *Neurips Workshop on Deep Learning*, 2011.

Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Yaofo Chen, Shijian Zheng, Peilin Zhao, and Mingkui Tan. Efficient test-time model adaptation without forgetting. In *International conference on machine learning*, pp. 16888–16905. PMLR, 2022.

Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, D Sculley, Sebastian Nowozin, Joshua V. Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift, 2019. URL `https://arxiv.org/abs/1906.02530`.

Anusri Pampari and Stefano Ermon. Unsupervised calibration under covariate shift. *arXiv preprint arXiv:2006.16405*, 2020.

Jaewoo Park, Jacky Chen Long Chai, Jaeho Yoon, and Andrew Beng Jin Teoh. Understanding the feature norm for out-of-distribution detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1557–1567, 2023.

Geoff Pleiss, Manish Raghavan, Felix Wu, Jon Kleinberg, and Kilian Q Weinberger. On fairness and calibration. *Advances in neural information processing systems*, 30, 2017.

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.

Rajat Raina, Alexis Battle, Honglak Lee, Benjamin Packer, and Andrew Y Ng. Self-taught learning: transfer learning from unlabeled data. In *Proceedings of the 24th international conference on Machine learning*, pp. 759–766, 2007.

Thomas Schlegl, Philipp Seeböck, Sebastian M. Waldstein, Ursula Schmidt-Erfurth, and Georg Langs. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. *CoRR*, abs/1703.05921, 2017. URL `http://arxiv.org/abs/1703.05921`.

Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. *Advances in neural information processing systems*, 30, 2017.

Samuel Stanton, Pavel Izmailov, Polina Kirichenko, Alexander A Alemi, and Andrew G Wilson. Does knowledge distillation really work? *Advances in Neural Information Processing Systems*, 34:6906–6919, 2021.

Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *CoRR*, abs/1905.11946, 2019. URL `http://arxiv.org/abs/1905.11946`.

Hui Tang, Ke Chen, and Kui Jia. Unsupervised domain adaptation via structurally regularized deep clustering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 8725–8735, 2020.

Surat Teerapittayanon, Bradley McDanel, and H. T. Kung. Branchynet: Fast inference via early exiting from deep neural networks. *CoRR*, abs/1709.01686, 2017. URL `http://arxiv.org/abs/1709.01686`.

Kai Tian, Shuigeng Zhou, Jianping Fan, and Jihong Guan. Learning competitive and discriminative reconstructions for anomaly detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 5167–5174, 2019.

Tijmen Tieleman, Geoffrey Hinton, et al. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

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*, pp. 5018–5027, 2017.

Yoav Wald, Amir Feder, Daniel Greenfeld, and Uri Shalit. On calibration and out-of-domain generalization. *Advances in neural information processing systems*, 34:2215–2227, 2021.

Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. *arXiv preprint arXiv:2006.10726*, 2020a.

Jindong Wang, Cuiling Lan, Chang Liu, Yidong Ouyang, Tao Qin, Wang Lu, Yiqiang Chen, Wenjun Zeng, and Philip Yu. Generalizing to unseen domains: A survey on domain generalization. *IEEE Transactions on Knowledge and Data Engineering*, 2022.

Ximei Wang, Mingsheng Long, Jianmin Wang, and Michael Jordan. Transferable calibration with lower bias and variance in domain adaptation. *Advances in Neural Information Processing Systems*, 33:19212–19223, 2020b.

Hongxin Wei, Renchunzi Xie, Hao Cheng, Lei Feng, Bo An, and Yixuan Li. Mitigating neural network overconfidence with logit normalization. In *International Conference on Machine Learning*, pp. 23631–23644. PMLR, 2022.

Xinheng Wu, Jie Lu, Zhen Fang, and Guangquan Zhang. Meta ood learning for continuously adaptive ood detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 19353–19364, 2023.

Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V Le. Self-training with noisy student improves imagenet classification. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10687–10698, 2020.

Yurong You, Cheng Perng Phoo, Katie Luo, Travis Zhang, Wei-Lun Chao, Bharath Hariharan, Mark Campbell, and Kilian Q Weinberger. Unsupervised adaptation from repeated traversals for autonomous driving. *Advances in Neural Information Processing Systems*, 35:27716–27729, 2022.

Houssam Zenati, Chuan Sheng Foo, Bruno Lecouat, Gaurav Manek, and Vijay Ramaseshan Chandrasekhar. Efficient gan-based anomaly detection, 2018. URL `https://arxiv.org/abs/1802.06222`.

Chang-Bin Zhang, Peng-Tao Jiang, Qibin Hou, Yunchao Wei, Qi Han, Zhen Li, and Ming-Ming Cheng. Delving deep into label smoothing. *IEEE Transactions on Image Processing*, 30:5984–5996, 2021. doi: 10.1109/tip.2021.3089942. URL `https://doi.org/10.1109%2Ftip.2021.3089942`.

Weicheng Zhu, Matan Leibovich, Sheng Liu, Sreyas Mohan, Aakash Kaku, Boyang Yu, Laure Zanna, Narges Razavian, and Carlos Fernandez-Granda. Deep probability estimation, 2022. URL `https://openreview.net/forum?id=hdSn_X7Hfvz`.

A Appendix

A.1 Experimental Setup Details

We used TensorFlow 2.9 Abadi et al. (2015) with Nvidia CUDNN version 11.3 on an RTX 3080 16GB laptop GPU with 32GB of system memory. All experiments are run three times using $\texttt{random\_seed} = \texttt{0}, \texttt{1}, \texttt{2}$, respectively.

1. PACS Li et al. (2017) has 4 domains: *pictures, art, cartoon, sketch* with 7 classes. All images are resized to $(227, 227, 3)$ and scaled between $[0, 255]$. Tested using LOO. For both MobileNet and EfficientNet: We set t_{min} and t_{max} parameters to 1.00 and 3.0 respectively.
2. HomeOffice Venkateswara et al. (2017) has 4 domains: *art, clipart, product, real* with 65 classes. All images are resized to $(128, 128, 3)$ and scaled between $[0, 255]$. Tested using LOO. MobileNet: We set t_{min} and t_{max} parameters to 1.20 and 2.75 respectively. EfficientNet: We set t_{min} and t_{max} parameters to 1.00 and 1.75 respectively.
3. Digits is a combination of 3 "numbers" datasets: USPS Hull (1994), MNIST LeCun et al. (2010), and SVHN Netzer et al. (2011). The images are resized to $(32, 32, 1)$ and scaled between $[0, 255]$. There are 10 classes. Tested using LOO by training on the source domains' training sets and adapting to target domain's test set. SmallCNN: We set t_{min} and t_{max} to 1.20 and 2.75, respectively.
4. TinyImageNet-C (TIN-C) Le & Yang (2015), has 15 domains with 200 classes. All images are resized to $(256, 256, 3)$ and scaled between $[0, 255]$. Backbones are trained on corruption-free (source) training set, adapted to and evaluated on corrupted (target) domains. For each target domain, there are 5 tiers of corruption. MobileNet: We set t_{min} and t_{max} parameters to 1.50 and 3.00 respectively. EfficientNet: We set t_{min} and t_{max} parameters to 1.00 and 2.50 respectively.

A.2 Hyperparameters

We do most initial training on the source domain using $\texttt{RMS\_Prop}(\texttt{lr} = \texttt{2e} - \texttt{4})$ Tieleman et al. (2012) to minimize cross-entropy loss for $\texttt{epochs} = \{\texttt{15}, \texttt{15}, \texttt{5}, \texttt{25}\}$ for each enumerated dataset, respectively. SmallCNN is compiled and initially trained with the Adam optimizer Kingma & Ba (2014) in lieu of RMSProp. We estimate roughly 1,000 hours of GPU usage at 130 watts of power to conduct our experiments. Note that MobileNet expects inputs to be prepossessed in a unique manner. We use Tensorflow's off-the-shelf pre-processing layer for MobileNet at the input. For the tests of statistical significance, we compared DEC with the second best algorithm for each experiment.

Note that the following are parameters for what we compare against. Further clarification on their meaning can be found in their respective works. For ETA, we set $\texttt{E\_0} = \texttt{0.4} \cdot \ln(\texttt{C})$, as this was their recommended value, and $\epsilon = \{\texttt{0.6}, \texttt{0.1}, \texttt{0.4}, \texttt{0.125}\}$ for each enumerated dataset, respectively. These ϵ values were empirically chosen to help their performance. For T3A, we set the number of supports to retain, $M = \infty$, as this provides the lowest calibration error. For SoTTA, we set $\rho = 0.05$, $\mathcal{C}_0 = \{0.33, 0.33, 0.33, 0.66\}$ for each dataset respectively to help their performance, and $N_{SoTTA} = 64$ as per their recommendations. We use the authors' recommended batch sizes for all techniques.

We selected hyper parameters either by the recommendation of the respective authors' or in order to improve performance with respect to calibration error. For example, for ETA we experimented with various values for ϵ in order to lower their ECE as much as possible per dataset (not per domain shift). A similar process was used for selecting $\mathcal{C}_0$ to aid SoTTA.

For our own hyper parameters, we started with $t_{min} = 1.0$ and $t_{max} = 3.0$, then tuned them as we did the other works. We want to emphasize the robustness of our algorithm to the selection of $t_{min/max}$. **Our hyperparameters are chosen for each dataset, not for individual domain shifts.** As shown in Tables 2 and 7, we achieve a statistically significant reduction in ECE and NLL despite using consistent hyperparameters across **15 domain shifts**. Our results in Table 1 further support our claims of robustness, showing a statistically significant improvement even with fixed hyperparameters across the four domain shifts. Another note is that although step 6 of DEC (Algorithm 2) shows the teacher model inferencing, we can actually just store and reuse the logits computed in line 1 of the `CCR` Algorithm 1. This saves us one forward pass computation. We use a batch size of 50 for our DEC for all experiments.

A.3 Further Discussion of Related Work

As stated previously, the focus of our work is on TTA algorithms. However, we would like to discuss some studies on OOD calibration to provide supplementary background. The authors of Wang et al. (2020b) and Gong et al. (2021) remark on poor calibration in the OOD setting, however, according to their algorithms, they require the unlabeled-domain-shifted observations **during training**. That is, their algorithm is not for the deployed setting like ours is. Another work, Wald et al. (2021), discusses proxies for determining if a model would be calibrated during deployment, before deployment. We recognize the relevance of these works but emphasize that they address different facets or settings.

Recent work by Press et al. (2024) offers a complementary view on the failure modes of entropy minimization under domain shift. Their study shows that while early adaptation clusters test embeddings near training data and boosts accuracy, continued optimization pushes embeddings away—leading to accuracy collapse. This biphasic behavior helps explain why TTA methods often degrade in calibration over time. While their focus is on representation drift and unsupervised accuracy estimation, our work highlights a parallel failure mode: over-adaptation also induces epistemic miscalibration.

A.4 Additional Experiments

Table 7: Comparison of Shannon entropy, ECE, and NLL averaged across the 15 domain shifts of TIN-C. We use the MobileNet backbone. Standard deviations across domains are shown as subscripts. This experiment highlights how excessive certainty (low Shannon entropy) correlates with sub-optimal calibration. **Our reduction in ECE and NLL is statistically significant** ($p < 0.01$) while maintaining competitive accuracy.

Algorithm	Shannon Entropy					ECE (lower is better)					NLL (lower is better)				
	Tier 1	**Tier 2**	**Tier 3**	**Tier 4**	**Tier 5**	**Tier 1**	**Tier 2**	**Tier 3**	**Tier 4**	**Tier 5**	**Tier 1**	**Tier 2**	**Tier 3**	**Tier 4**	**Tier 5**
No Adapt	$1.44_{0.10}$	$1.53_{0.14}$	$1.61_{0.18}$	$1.67_{0.19}$	$1.66_{0.21}$	$0.43_{0.02}$	$0.44_{0.03}$	$0.47_{0.05}$	$0.50_{0.06}$	$0.53_{0.08}$	$5.98_{0.55}$	$6.56_{0.88}$	$7.36_{1.41}$	$8.42_{2.04}$	$9.50_{2.66}$
DEC (ours)	$3.35_{0.21}$	$3.55_{0.30}$	$3.80_{0.42}$	$3.97_{0.42}$	$4.04_{0.38}$	$\mathbf{0.05}_{0.01}$	$\mathbf{0.05}_{0.02}$	$\mathbf{0.06}_{0.02}$	$\mathbf{0.07}_{0.03}$	$\mathbf{0.10}_{0.06}$	$\mathbf{2.68}_{0.11}$	$\mathbf{2.83}_{0.17}$	$\mathbf{3.05}_{0.29}$	$\mathbf{3.37}_{0.46}$	$\mathbf{3.78}_{0.78}$
T3A	$1.60_{0.10}$	$1.70_{0.14}$	$1.81_{0.17}$	$1.87_{0.17}$	$1.89_{0.21}$	$0.42_{0.02}$	$0.44_{0.02}$	$0.46_{0.03}$	$0.48_{0.05}$	$0.50_{0.06}$	$6.36_{0.66}$	$7.15_{1.10}$	$9.17_{3.37}$	$13.27_{7.59}$	$19.55_{15.24}$
ETA	$1.19_{0.10}$	$1.22_{0.10}$	$1.36_{0.15}$	$1.32_{0.19}$	$1.34_{0.16}$	$0.51_{0.04}$	$0.54_{0.05}$	$0.56_{0.05}$	$0.59_{0.06}$	$0.62_{0.06}$	$8.69_{0.98}$	$8.93_{0.86}$	$9.83_{2.17}$	$10.68_{2.02}$	$12.06_{2.42}$
TENT	$1.10_{0.04}$	$1.14_{0.06}$	$1.21_{0.09}$	$1.28_{0.12}$	$1.34_{0.12}$	$0.38_{0.02}$	$0.40_{0.03}$	$0.43_{0.04}$	$0.47_{0.06}$	$0.50_{0.07}$	$4.95_{0.35}$	$5.47_{0.63}$	$6.25_{1.17}$	$7.35_{1.94}$	$8.54_{2.83}$
SoTTA	$1.14_{0.04}$	$1.19_{0.05}$	$1.26_{0.09}$	$1.34_{0.12}$	$1.43_{0.16}$	$0.36_{0.01}$	$0.37_{0.02}$	$0.39_{0.03}$	$0.42_{0.04}$	$0.45_{0.05}$	$4.70_{0.23}$	$4.99_{0.37}$	$5.44_{0.66}$	$6.10_{1.01}$	$6.89_{1.53}$

Table 8: Accuracy on TIN-C with EfficientNet backbone across different tiers of corruption for various algorithms. Standard deviations are shown as subscripts.

Algorithm	**Tier 1**	**Tier 2**	**Tier 3**	**Tier 4**	**Tier 5**
No Adapt	$0.41_{0.04}$	$0.37_{0.06}$	$0.31_{0.08}$	$0.24_{0.09}$	$0.19_{0.10}$
DEC (ours)	$\mathbf{0.49}_{0.02}$	$\mathbf{0.46}_{0.04}$	$0.41_{0.06}$	$0.35_{0.10}$	$0.29_{0.11}$
T3A	$0.42_{0.04}$	$0.38_{0.06}$	$0.32_{0.08}$	$0.26_{0.10}$	$0.21_{0.10}$
ETA	$0.33_{0.03}$	$0.29_{0.06}$	$0.26_{0.07}$	$0.22_{0.08}$	$0.19_{0.09}$
TENT	$0.47_{0.03}$	$0.42_{0.05}$	$0.37_{0.08}$	$0.31_{0.10}$	$0.25_{0.11}$
SoTTA	$\mathbf{0.49}_{0.02}$	$\mathbf{0.46}_{0.03}$	$\mathbf{0.42}_{0.06}$	$\mathbf{0.36}_{0.09}$	$\mathbf{0.31}_{0.10}$

A.5 Run-to-Run Variances across all TTA algorithms

- For the Digits dataset: $\sigma^2_{\mathtt{max}} = [4.39 \times 10^{-3}, 5.51 \times 10^{-3}, 3.85 \times 10^{-3}, 1.00 \times 10^{-3}]$ for accuracy, ECE, and entropy, respectively across all trials using SmallCNN.

Table 9: Average accuracy, ECE, entropy, and NLL on the Home Office dataset using the EfficientNet backbone. Domain-to-domain $\sigma^2_{\max}$ values for accuracy, ECE, entropy, and NLL are reported.

Algorithm	**Accuracy**	**ECE**	**Entropy**	**NLL**
No Adapt	0.665	0.198	0.642	2.184
DEC	0.669	**0.069**	1.832	**1.490**
T3A	**0.686**	0.267	0.186	4.321
ETA	0.665	0.218	0.422	2.745
TENT	0.684	0.211	0.460	2.396
SoTTA	0.667	0.194	0.648	2.097
$\sigma^2_{\max}$	0.060	0.030	1.110	3.760

Table 10: Average accuracy, ECE, entropy, and NLL on the Home Office dataset using the MobileNet backbone. Domain-to-domain $\sigma^2_{\max}$ values for accuracy, ECE, entropy, and NLL are reported.

Algorithm	**Accuracy**	**ECE**	**Entropy**	**NLL**
No Adapt	0.567	0.242	0.824	2.184
DEC	0.575	**0.046**	2.183	**1.686**
T3A	**0.617**	0.335	0.182	4.321
ETA	0.553	0.254	0.807	2.745
TENT	0.579	0.242	0.776	2.441
SoTTA	0.574	0.233	0.833	2.409
$\sigma^2_{\max}$	0.013	0.011	0.116	6.467

- For Home Office across both backbones: $\sigma^2_{\texttt{max}} = [5.00 \times 10^{-2}, 1.12 \times 10^{-5}, 1.66 \times 10^{-5}, 7.29 \times 10^{-3}]$ for accuracy, ECE, and entropy, respectively.
- For the PACS dataset across both backbones: $\sigma^2_{\texttt{max}} = [1.21 \times 10^{-3}, 1.02 \times 10^{-4}, 2.09 \times 10^{-3}, 2.00 \times 10^{-4}]$ for accuracy, ECE, and entropy, respectively across all trials.
- For the TIN-C dataset across both backbones and across all 5 domain shifts: $\sigma^2_{\texttt{max}} = [1.39 \times 10^{-3}, 1.12 \times 10^{-7}, 2.46 \times 10^{-3}, 2.40 \times 10^{-1}]$ for accuracy, ECE, and entropy, respectively across all trials.

Table 11: Average accuracy, ECE, entropy, and NLL on the PACS dataset using the MobileNet backbone. Domain-to-domain $\sigma^2_{\max}$ values for accuracy, ECE, entropy, and NLL are reported.

Algorithm	**Accuracy**	**ECE**	**Entropy**	**NLL**
No Adapt	0.841	0.111	0.177	1.182
DEC (ours)	0.848	**0.082**	0.341	**0.466**
T3A	**0.857**	0.116	0.096	0.866
ETA	0.842	0.101	0.192	1.213
TENT	0.848	0.108	0.162	1.160
SoTTA	0.852	0.102	0.180	1.188
$\sigma^2_{\max}$	0.034	0.015	0.099	1.016

A.6 SmallCNN

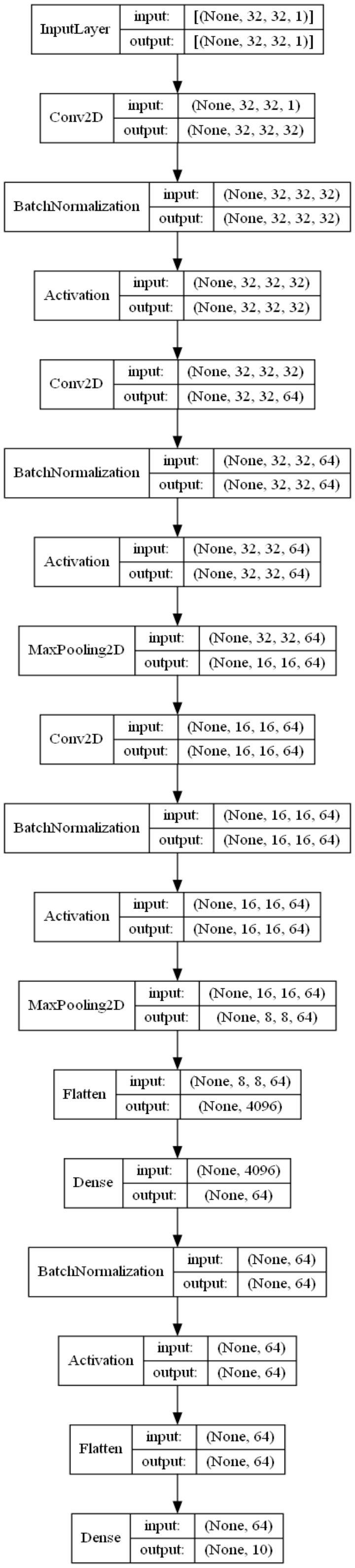
InputLayer input: [(None, 32, 32, 1)] output: [(None, 32, 32, 1)]
Conv2D input: (None, 32, 32, 1) output: (None, 32, 32, 32)
BatchNormalization input: (None, 32, 32, 32) output: (None, 32, 32, 32)
Activation input: (None, 32, 32, 32) output: (None, 32, 32, 32)
Conv2D input: (None, 32, 32, 32) output: (None, 32, 32, 64)
BatchNormalization input: (None, 32, 32, 64) output: (None, 32, 32, 64)
Activation input: (None, 32, 32, 64) output: (None, 32, 32, 64)
MaxPooling2D input: (None, 32, 32, 64) output: (None, 16, 16, 64)
Conv2D input: (None, 16, 16, 64) output: (None, 16, 16, 64)
BatchNormalization input: (None, 16, 16, 64) output: (None, 16, 16, 64)
Activation input: (None, 16, 16, 64) output: (None, 16, 16, 64)
MaxPooling2D input: (None, 16, 16, 64) output: (None, 8, 8, 64)
Flatten input: (None, 8, 8, 64) output: (None, 4096)
Dense input: (None, 4096) output: (None, 64)
BatchNormalization input: (None, 64) output: (None, 64)
Activation input: (None, 64) output: (None, 64)
Flatten input: (None, 64) output: (None, 64)
Dense input: (None, 64) output: (None, 10)

Figure 4: SmallCNN’s architecture has 319,498 parameters.