Have You Already Tried Turning Your Model Off And On Again? Towards Stable Continual Test-Time Adaptation

Anonymous Author(s)
Affiliation
Address
email

Abstract

Recently, many test-time adaptation methods have shown impressive results in adapting to image corruptions on ImageNet scale. These methods are increasingly benchmarked on continual distribution shifts that are ubiquitous under realistic conditions. Here, we examine these benchmarks, and show that they suffer from two problems: they are not long enough to reveal the asymptotic behaviour of methods, and their difficulty fluctuates wildly throughout, making them hard to understand. To remedy these issues, we propose *Continually Changing Corruptions* (CCC), a benchmark that allows for virtually infinite testing and is stable in terms of difficulty. Using this benchmark, we show that eventually all but one state-of-the-art methods collapse to below baseline accuracy, including two methods with built-in anti-collapse mechanisms. Additionally, we show that a simple baseline approach achieves state-of-the-art results on established benchmarks as well as our proposed benchmark, all without actually accumulating knowledge over time. Our approach calls into question the amount of progress made since [43] on this task.

1 Introduction

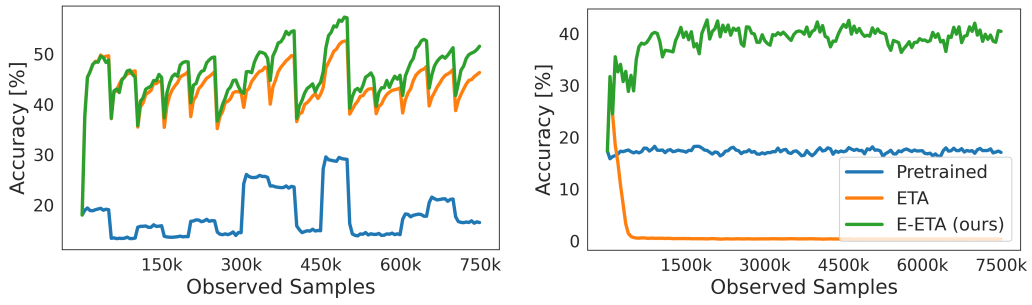
Traditionally, machine learning assumes training and test data to be i.i.d and hence relatively small fixed datasets are used to evaluate the test performance of such algorithms. However, if a model is going to be used perpetually for a long time in the world, it is crucial to ensure stable performance under continuously changing conditions (as caused e.g. by weather, lighting or sensory hardware degradation). Test-time adaptation methods have become increasingly popular for adapting trained computer vision models to distribution shifts and drifts at test-time.

To reach robustness on ImageNet scale classification [35], previous techniques include pre-training of large models on diverse datasets [23, 31, 46] and robustification of smaller models by specifically designed data augmentation [12, 34]. Recent works in this domain [8, 25, 26, 27, 33, 36, 43, 44] have been dominated by one approach: test time adaptation (TTA). In TTA, a model exclusively uses incoming test data to continuously adapt. Even though other approaches alter the training procedure, or use both train and test data, TTA, using only test data, and without altering training, has been shown to be superior [26, 36, 43].

TTA approaches take in a batch of unlabeled inputs and optimize an unsupervised loss on those inputs, in order to improve classification accuracy. In our setting, we consider the ImageNet classification task with 1,000 classes. Previous TTA work [8, 25, 26, 33, 36, 43] evaluate their models on ImageNet-C or smaller scale image classification benchmarks. ImageNet-C consists of 75 copies of the ImageNet

Code available at: `https://github.com/oripress/CCC`

Submitted to 37th Conference on Neural Information Processing Systems (NeurIPS 2023). Do not distribute.



(a) (b)

Figure 1: **(a)** CIN-C, the current benchmark features abrupt changes between noises, which causes the baseline accuracy (blue) to vary wildly. **(b)** CCC, our proposed benchmark is 10 times longer, more diverse, and features continuous changes at a steady baseline accuracy. Crucially, CCC is able to show that methods like ETA collapse, while CCC is not. In both cases, the average accuracy of the baseline is nearly identical.

validation set, wherein each copy is corrupted according to 15 different noises at 5 different severity levels. When TTA models are evaluated on ImageNet-C, they are adapted on each noise and severity combination individually starting from their pretrained weights.

However, the data distribution changes not only when a trained model first encounters real data under deployment condition. Under realistic conditions, the test data changes all the time, e.g., due to weather changes. TTA methods are by design readily applicable to this settings and recently the field has started to move towards testing TTA models under continual adaptation settings [7, 27, 30, 40, 44]. Strikingly, this revealed that the dominant TTA approach Tent [43] decreases in accuracy over time, eventually being less accurate than a non-adapting baseline [27, 44].

This collapsing behaviour of Tent demonstrates that it is crucial to subject TTA methods to continual domain change conditions, long enough to reveal the asymptotic behaviour. While previous benchmarking of TTA methods already managed to reveal the collapse of Tent, our work will show that in fact all TTA methods collapse sooner or later, *including methods with explicit built-in anti-collapse strategies*. Furthermore, additional confounders, such as the difficulty of the images themselves, make it hard to understand whether a model's accuracy has gone down as a result of collapse, or simply because the current noise it is adapting to is more difficult.

To remedy these issues, we introduce an image classification benchmark designed to thoroughly evaluate TTA models, while taking into account factors that can affect adaptation. Our benchmark, *Continuously Changing Corruptions* (CCC), tests models for their ability to adapt to image corruptions that are constantly changing, much like when fog turns to rain or day turns to night. CCC allows us to easily control different factors that could affect the ability of a given method to continuously adapt: the corruptions and their order, the difficulty of the images themselves, and the speed at which corruptions transition. Importantly, the length of our benchmark is ten times larger than that of previous benchmarks, while also being much more diverse. Using CCC, we discover that seven recently published state-of-the-art TTA methods are less accurate than a non-adapting, pretrained model. While Tent was already shown to collapse [7, 27, 44], we show that this problem is not specific to Tent but many other methods – including specifically designed continual adaptation methods – collapse as well.

Finally, we show how simply resetting the model to its pretrained weights every 1,000 steps can counteract collapse. Although previous works employ methods that accumulate knowledge over time, we show that a constantly resetting baseline method is better on both existing benchmarks and ours (CCC). In addition to our baseline method being much simpler than previous work, it is also faster to run, and requires less hyperparameters.

In summary, our work makes the following important points:

- We show the downsides of current continual adaptation benchmarks: they are too short to reveal asymptotic behaviour and too uncontrolled to show short-term dynamics.

- We introduce CCC, a continual adaptation benchmark that fixes the aforementioned downsides. Using CCC, we show that all current TTA methods fail over time and become worse than a non-adapting, pretrained baseline.
- We show that a baseline method, that does not accumulate knowledge over long periods of time, currently sets the state of the art across current and proposed benchmarks, which casts doubt on recent progress made in TTA [29]

2 Towards Infinite Testing

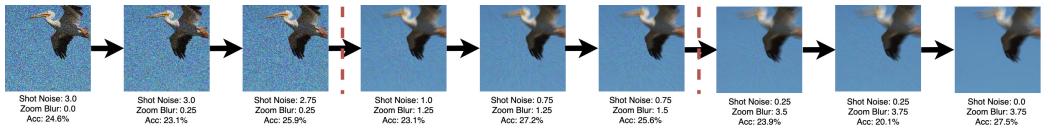


Figure 2: CCC enables smooth transitions between noises, while staying close to any desired baseline accuracy (25% in this case). Zoom in for more detail.

For most of the time, it was common to evaluate TTA methods only on datasets with singular domain shifts such as the individual corruptions of ImageNet-C [14]. However, the world is steadily changing and recently the community started moving towards continual adaptation, i.e., evaluating methods with respect to their ability to adapt to ongoing domain shifts [27, 30, 40, 44].

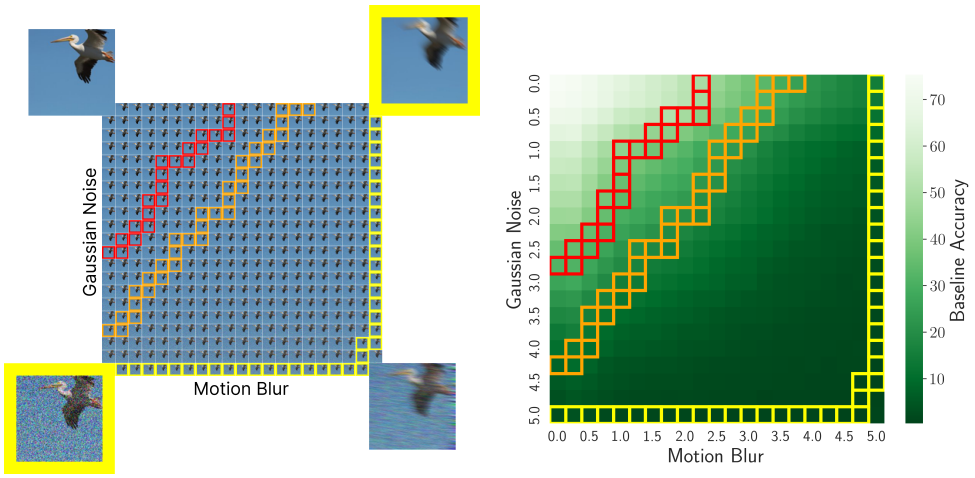
The dominant method of evaluating continual adaptation on ImageNet scale is to concatenate the top severity datasets of the 15 ImageNet-C corruptions into one big-dataset. We refer to the variant of this dataset introduced by [44] as *Concatenated ImageNet-C* (CIN-C) in the following. It was used by [7, 27, 44] to demonstrate the collapse of Tent and the stability of their respective methods. In Figure 5a, we evaluate a range of TTA methods on CIN-C and notice three potential problems: **First**, while some methods clearly collapse to chance performance (Tent, RPL, CPL), for other methods the dataset indicates a downwards trend, but is not long enough to see the asymptotic behavior. For example, the performance of CoTTA clearly goes down, but it's not yet clear whether it approaches chance performance as in the case of Tent, or stabilizes above or below (such as SLR) baseline performance. **Secondly**, assessing adaptation dynamics is further complicated by the fact that baseline performance exhibits significant variations among the different corruptions in CIN-C. This leads to substantial fluctuations in performance across multiple runs, making it difficult to obtain a clear and reliable assessment. **Lastly**, CIN-C features exclusively hard transitions between different corruption types. However, in the real world, most domain changes are smooth with varying speeds: day to night, rain to sunshine, or the accumulation of dust on a camera.

To address these issues, we propose a new benchmark, *Continuously Changing Corruptions* (CCC). CCC solves the issues of length, uncontrolled baseline difficulty, and transition smoothness in a simple and effective manner. The length issue is remedied because the individual runs of CCC are constructed by a generation process which can generate very long datasets without reusing images. In this work we use runs of 7.5M images, which is 10 times as long as CIN-C. Since both [27, 43] have shown that dataset difficulty is a confounder in adaptation, the difficulty of individual benchmark runs is kept stable. Additionally, we examine three different difficulty levels to ensure a comprehensive evaluation. Furthermore, CCC exhibits smooth domain shifts: it transitions in a continual way from one corruption to the next by applying two corruptions two each image such that the severity of the first corruption can be smoothly decreased while the severity of the second corruption increases. Finally, our benchmark also takes into account the speed at which noise types transition, a factor not considered in previous benchmarks. CCC has three difficulty levels: CCC-Easy, CCC-Medium, and CCC-Hard, with baseline difficulties of 34%, 17%, and 2%, respectively. We will now outline the generation procedure of the dataset.

Image Corruptions One of our desiderata is that we want to be able to keep the baseline accuracy constant over time. Since the difficulty of the ImageNet-C corruptions varies greatly from severity to severity, and also from corruption to corruption when keeping severity constant, we first introduce more fine-grained severity levels to ImageNet-C: We define severities from 0 to 5 in steps of 0.25 by interpolating the parameters of the original implementation functions. In order to allow for smooth transitions between corruptions, we will furthermore need to apply two different ImageNet-C

corruptions to each image, such that we can decrease the severity of one corruption while increasing the severity of another one. Hence, the corruptions of CCC are given by quadruples (n_1, s_1, n_2, s_2), where n_1 and n_2 are ImageNet-C corruption types and s_1 and s_2 are severity levels ranging from 0 to 5 in steps of 0.25. When applying such a corruption, we first apply n_1 and then n_2 at their respective severities (see Figure 3a).

Calibration For controlling baseline accuracy, we need to know how difficult each combination of 2 noises and their respective severities is. To that end, we first select a subset of 5,000 images from the ImageNet validation set. For each corruption (n_1, s_1, n_2, s_2), we corrupt all 5,000 images accordingly and evaluate the resulting images with a pre-trained ResNet-50 [10]. The resulting accuracy is what we refer to as *baseline accuracy* and what we use for controlling difficulty.



(a) (b)

Figure 3: **(a):** Each corruption of CCC consists of applying two ImageNet-C corruptions at different severities. We extend the individual severities to be more fine-grained than in ImageNet-C, allowing for smoother noise changes. The corners are enlarged for easier viewing, zoom in for greater detail. **(b):** Sample dataset sequences with a constant baseline accuracy. The sequences start from the left where Motion Blur is zeroed out, and end at the top with Gaussian noise zeroed out. The colors red, orange, and yellow correspond to target baseline accuracies of 34%, 17%, and 2%, respectively. The x and y axes indicate noise severity levels.

Generating Benchmark Runs Given the calibration, we prepare benchmark runs with different baseline accuracies, transition speeds, and noise orderings. We pick 3 different baseline accuracies: 34%, 17%, and 2% (CCC-Easy, CCC-Medium, CCC-Hard respectively). For each one of the difficulties, we select a further 3 transition speeds: 1k, 2k, 5k. Lastly, for each difficulty and transition speed combination we use 3 different noise orderings, determined by 3 random seeds. To generate each run, we first select the initial corruption at the severity which according to our calibration is closest to the desired baseline accuracy. We then transition to the second corruption of the noise ordering by repeatedly either decreasing the severity of the first noise by 0.25 or increasing the severity of the second noise by 0.25 such that the baseline accuracy is as close to the target as possible (see Figure 3). In each step along each path, we sample 1k, 2k, or 5k images from the ImageNet validation set depending on the desired transition speed. Each image is randomly cropped and flipped for increasing the diversity of the dataset, and then corrupted.

Once the path from the initial to the second corruption is finished, the process is repeated for transitioning to the third corruption and so on (for more details see Appendix B). In the end, we have 3 difficulties consisting of 9 benchmark runs each. CCC-Medium at a speed of 2k corresponds roughly to CIN-C's difficulty and transition speed.

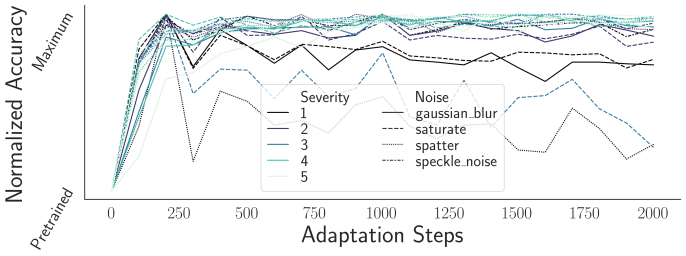


Figure 4: ETA accuracy over time, on the ImageNet-C holdout noises and each of their severities. For almost every noise in the holdout set, ETA reaches its maximum accuracy very quickly, within 200-300 steps.

3 Episodic resetting for continual adaptation

Using CCC, we are able to test how previous work holds up under various conditions, at long time scales. We discover that none of the previous methods reliably stop collapse. Hence, we propose a very simple baseline method to remedy this: Simply resetting the model after a fixed number of adaptation steps. Surprisingly, this approach has never been tried before [1]

3.1 Resetting vs other anti-collapse strategies

Typically, previous works optimize two losses: one loss encourages adaptation on unseen data, another loss regularizes the model to prevent collapse. Intuitively, having to optimize two losses should be harder than optimizing just one. We therefore hypothesize that optimizing an adaptation loss and an anti-collapse loss could yield worse performance than just optimizing an adaptation loss alone, at least in the short term (before the collapse occurs).

Empirically, we see evidence for this in short term adaptation on CIN-C: ETA and EATA optimize the same loss, but EATA additionally optimizes an anti-collapse loss. Consequently, ETA beats EATA by 2% on CIN-C (however this does not hold for longer benchmarks, such as CCC, where ETA collapses). Additionally, ETA is quick to adapt to new noises from scratch. On each of the holdout set noises and severities, ETA reaches its maximum accuracy after around 200 adaptation steps, see Figure 4.

Given these two observations, namely **(1)** optimizing an anti-collapse loss alongside the main loss can lead to reduced accuracy in short term adaptation and **(2)** ETA reaches max adaptation accuracy quickly, we decide to test a different approach: instead of trying to fight collapse like in [25, 27, 44], we can instead simply reset the model to its initial weights every so often, and start the adaptation anew.

We build on the ETA method, which currently sets the state-of-the-art on CIN-C and is a version of EATA [27], just without an anti-collapse loss. The exact objective function is outlined in Appendix C. To determine how long the model should run before being reset, we used the holdout noises of Imagenet-C, concatenated together like in CIN-C (Table 1).

Table 1: Accuracy of our method for different resetting values on CIN-C holdout noises.

k (steps)	125	250	500	1000	1500	2000
Acc. [%]	42.1	44.4	46.0	**46.7**	46.5	46.4

4 Experiment Setup

We benchmark against a range of published TTA models. For all models, we use a batch size of 64. In all models, the BatchNorm statistics are estimated on the fly, and the affine shift and scale parameters are optimized according to some model-specific strategy.

[1] We elaborate on this in Appendix E.

- **BatchNorm (BN)** [26, 36] estimates the BatchNorm statistics (mean and variance) separately for each batch at test time. The affine transformation parameters are not adapted at all.
- **Tent** [42] optimizes the entropy objective on the test set in order to update the scale and shift parameters of batchnorm (in addition to learning the statistics).
- **Robust Pseudo-Labeling (RPL)** [33] uses a teacher-student approach in combination with a label noise resistant loss.
- **Conjugate Pseudo Labels (CPL)** [8] use meta learning to learn the optimal adaptation objective function across a class of possible functions
- **SLR** [25] uses a loss function that is similar to entropy, but without vanishing gradients. *Anti-Collapse Mechanism:* An additional loss is used to encourage the model to have uniform predictions over the classes, and the last layer of the network is kept frozen.
- **Continual Test Time Adaptation (CoTTA)** [44] uses a teacher student approach in combination with augmentations. *Anti-Collapse Mechanism:* Every iteration, 1% of the weights are reset back to their pretrained values.
- **Efficient Test Time Adaptation (EATA)** [27] uses 2 weighing functions to weigh its outputs: the first based on their entropy (lower entropy outputs get a higher weight), the second based on diversity (outputs that are similar to seen before outputs are excluded). *Anti-Collapse Mechanism:* An L_2 regularizer loss is used to encourage the model's weights to stay close to their initial values.
- **EATA Without Weight Regularization (ETA)** For completeness, we also test ETA, which is EATA but without the regularizer loss, proposed in [27].

Following the original implementations, Tent, ETA, and EATA use SGD with a learning rate of $2.5 \cdot 10^{-5}$. RPL uses SGD with a learning rate of $5 \cdot 10^{-3}$. SLR uses the Adam optimizer with a learning rate of $6 \cdot 10^{-4}$. CoTTA uses SGD with a learning rate of 0.01, and CPL uses SGD with a learning rate of 0.001.

Optimal reset interval To determine the optimal reset interval, we run ETA with reset intervals $k \in [125, 250, 500, 1000, 1500, 2000]$ on CIN-C using the IN-C holdout noises, following the recommendations for TTA hyperparameter tuning given in Rusak et al. [33]. We the 4 holdout noises at severity 5 as our base test set. This base test set is repeated until the model sees 750k images, which is equal to the length of CIN-C. We do this for every permutation of the 4 holdout corruptions (see Table 1). On this holdout set, we find that the optimal k is equal to 1,000.

5 Results

Episodic resetting is a strong baseline for continual adaptation. Our method outperforms all previous methods on both established benchmarks (CIN-C, CIN-3DCC) and our continual adaptation benchmark, CCC (Table 2 and Figure 5). Episodic resetting outperforms both the (gradient-free) adapting BatchNorm adaptation we run as a control, as well as the ETA baseline we base our method on. We also outperform the previous state-of-the-art, EATA and increase accuracy by more than 11% on CIN-C (improving from 41.8% to 46.5%), and by almost 8% when averaged across CIN-C, and all variants of CCC.

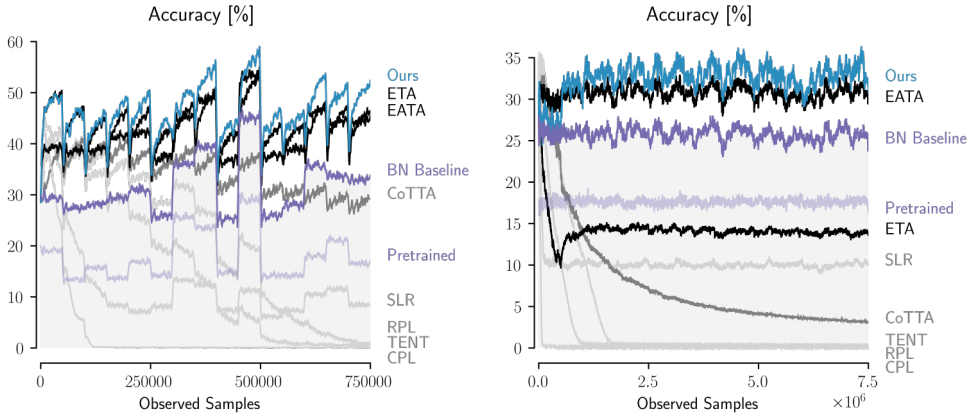
We also observe the need for long adaptation benchmarks: While multiple methods (SLR, CPL, TENT, RPL) already collapse below performance of a pre-trained network or a BatchNorm baseline on CIN-C, the collapse of CoTTA and ETA is not visible on this dataset. Benchmarking these methods across different levels of difficulty, and on longer timescales is necessary to observe their collapse: CoTTA collapses on all CCC variants, ETA collapses on CCC-Medium and CCC-Hard.

The results transfer to Vision Transformers. To further validate our claims, we test both EATA and our method when using a Vision Transformer [3] backbone. The difference in average accuracy between our method and EATA is larger when using a ViT, as compared to a ResNet-50: on CIN-C and CCC the gap is 10.9% and 11.7% respectively. Additionally, EATA's accuracy on CCC is below that of a pretrained, non-adapting model [1]. This collapse can only be seen by using CCC, and not when evaluating on CIN-C.

[1] Increasing the regularizer parameter value does not help stablize the model, see Appendix D

Table 2: Mean accuracy of ResNet-50 models on both CCC and CIN-C. For each CCC split, a mean of 9 runs is taken. For the CIN-C experiments the accuracy shown is the mean of 10 different noise permutations. We outperform both our Tent (gradient-based adaptation, instable) and BatchNorm (gradient-free adaptation, stable) baselines and achieve SoTA performance compared to previous models.

Adaptation method	CIN-C	CCC-Easy	CCC-Medium	CCC-Hard	Average
Pre-trained [10]	18.0	34.1	17.3	1.5	17.7
Tent [43]	15.6	3.9	1.4	0.51	5.4
RPL [33]	21.8	7.5	2.7	0.67	8.2
SLR [25]	12.4	22.2	7.7	0.66	8.2
CPL [8]	3.0	0.41	0.22	0.14	0.94
CoTTA [44]	34.0	14.9	7.7	1.1	14.4
EATA [27]	41.8	48.2	35.4	8.7	33.5
BN (control)	31.5	42.6	27.9	6.8	27.2
ETA (baseline)	43.8	41.4	1.1	0.23	21.6
Ours (E-ETA)	**46.5**	**49.3**	**38.9**	**9.6**	**36.1**

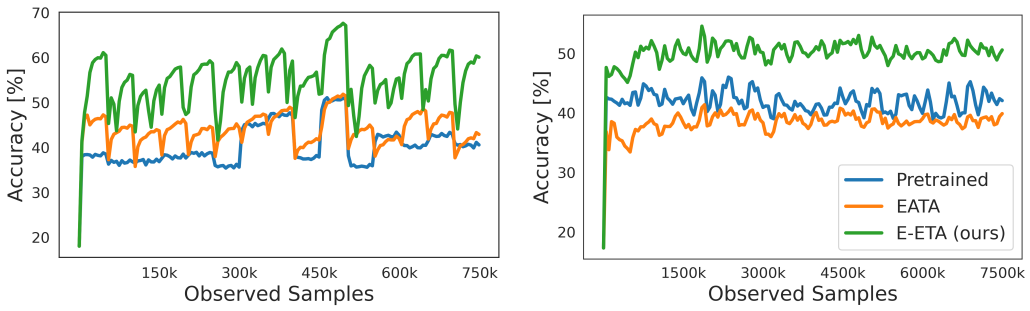


(a) (b)

Figure 5: Adaptation performance of all evaluated models depending on the number of observed samples so far. (a) CIN-C. Model performances are averaged over the 10 runs of the benchmark. (b) CCC. Model performances are averaged over the 27 runs of the three difficulty levels. See Appendix A, Figure 7 for separate plots for CCC Easy, Medium and Hard.

CCC is more insightful than CIN-C. For three models tested, CIN-C is either inconclusive or wrong: For CoTTA, CIN-C shows it to be on a downward trend, but still higher than a pre-trained baseline (Figure 5a). Additionally, ETA shows no signs of collapse on CIN-C, while collapsing very clearly on CCC (Figure 5b, more precisely on CCC-Medium and CCC-Hard, see Appendix Figure 7). When tested using a ViT backbone, EATA is better than the pretrained baseline on CIN-C, but worse on CCC. Lastly, SLR on CIN-C appears to be somewhat stable (ableit at around 10% accuracy). CCC reveals this to be only partly true: on CCC-Hard, SLR is not stable and collapses to nearly chance accuracy. In summary, models evaluated on CCC show clear limits, which are impossible to see on CIN-C because of the high difficulty variance between runs, and its short length.

Episodic resetting is less sensitive to hyperparameters An added benefit to our method is that it is less sensitive to hyperparameters than EATA. We conduct a simple hyperparam search of the E_0 parameter - the hyperparameter that controls how many outputs get filtered out because of their high entropy. Our method consistently outperforms EATA across all but one value, and for the highest value, 0.7, EATA collapses to almost chance accuracy on all splits, while our method does not. In addition, our method's performance benefits from finetuning ($E_0 = \{0.2, 0.3\}$), while EATA is not able to improve.



(a) (b)

Figure 6: Using a ViT backbone: **(a)** On CIN-C, EATA is 4% better than the baseline. **(b)** On CCC-Medium, EATA is 3.6% worse than the pretrained baseline. E-ETA (ours) is consistently better than both EATA and the baseline.

Table 3: Average accuracy on all of CCC splits on a variety of E_0 values. For all experiments in this paper we use $E_0 = 0.4 \times \ln(10^3)$, as in [27].

$E_0 \times \ln(10^3)$	0.1	0.2	0.3	0.4	0.5	0.6	0.7
EATA	27.8	27.9	29.9	**30.8**	28.7	28.0	0.33
Ours	31.6	32.9	**33.1**	32.6	30.7	25.7	16.8

The results transfer to Imagenet-3D Common Corruptions To further demonstrate the effectiveness of our method, we show results on Imagenet-3DCC (IN-3DCC)[17], which features 12 types of corruptions, Similarly to CIN-C, we test our models on 10 different permutations of concatenations of all the noises of IN-3DCC, which we call CIN-3DCC. As in the case of CIN-C and CCC, our method outperforms all previous methods (Table 4).

Table 4: Mean accuracy of ResNet-50 models on CIN-3DCC, over 10 random permutations of the noises.

Adaptation technique	CIN-3DCC
Pre-trained [10]	31.5
Tent [43]	24.4
RPL [33]	30.0
SLR [25]	12.2
CPL [8]	5.7
CoTTA [44]	37.6
EATA [27]	43.6
BN (control)	35.7
ETA (baseline)	42.7
Ours	**45.2**

CCC doesn't necessarily require a lot of adaptation steps to be insightful.

Even though CCC's insights are shown to be more reliable than CIN-C's when it comes to model collapse, CCC is much larger. We're interested in how adaptation looks like asymptotically, and we see that it only takes around 15k adaptation steps (slightly longer than CIN-C) for methods to reach their limit on CCC-Hard. This means that the first 15k adaptation steps of a model on CCC-Hard can be used as a quick test to determine whether a model collapses or not, which is already much more reliable than CIN-C (see Figure 7).

6 Discussion and Related Work

6.1 Domain Adaptation

In practice, the data distribution at deployment is different from training, and hence the task of *domain adaptation*, i.e., the task of adapting models to different target distributions has received a lot of attention [6, 15, 19, 20, 34, 40, 43]. The methods on domain adaptation split into different categories based on what information is assumed to be available during adaptation. While some methods assume access to labeled data for the target distribution [24, 48], *unsupervised domain adaptation* methods assume that the model has access to labeled source data and unlabeled target data at adaptation time [5, 19, 20, 39]. Most useful for practical applications is the case of *test-time adaptation*, where the task is to adapt to the target data on the fly, without having access to the full target distribution, or the original training distribution [26, 27, 33, 36, 40, 43, 49].

In addition to the division made above, one can further distinguish what assumptions are made about how the target domain is changing. Many academic benchmarks are focusing on one-time distribution shifts. However, in practical applications, the target distribution can easily change perpetually over time, e.g., due to changing weather and lightness conditions, or due to sensor corruptions. Therefore, the latter setting of *continual adaptation* has been receiving increasing attention recently. The earliest example of adapting a classifier to an evolving target domain that we are aware of is [16], which learn a series of transformations to keep the data representation similar over time. [5, 41, 45] use an adversarial domain adaptation approach for this. [2] pointed out that two of these approaches can be prone to catastrophic forgetting [32]. To deal with this, different solutions have been proposed [1, 2, 21, 25, 27, 44].

Test-time adaptation methods have classically been applied in the setting of one-time domain change, but can be readily applied in the setting of continual adaptation, and some recent methods have been explicitly designed and tested with continual adaptation in mind [27, 43, 44]. Because TTA methods use only test data and don't alter the training procedure, they are particularly easy to apply and have been shown to be superior to other domain adaptation approaches [6, 26, 34, 36], Therefore, we here focus on TTA methods only, which we discussed in more detail in Section 4.

6.2 Continual Adaptation Benchmarks

While continually changing datasets are used in the continual learning literature, e.g. [4, 9, 22, 37, 38, 47], it has been used in TTA benchmarks only very recently. In contrast to all previous benchmarks, we want to evaluate how continual adaptation methods change over long periods of time, when the noise changes in a continuous manner. The longest datasets for TTA were made up of hundreds of thousands of labeled images in total, while we adapt to 7.5M images per run. Other datasets are comprised of short video clips [22, 37, 38] 10-20 seconds in length. Additionally, with our noise synthesis, we can guarantee a wide variety of noises in each one of our evaluation runs, we can control the speed at which the noise changes, and we can control the difficulty of the generated noise. Lastly, CCC takes into account different adaptation speeds: [33] and [25] both show that running their methods on more than 1 epoch of ImageNet-C improves performance. By controlling the transition speed, we can more thoroughly evaluate how models adapt to different scenarios.

7 Conclusion

We proposed CCC, a benchmark made specifically for rigorously testing continually adapting methods. By using CCC, we discovered that seven popular test time adaptation methods can collapse to less than a non-adapting, pretrained classifier's accuracy during adaptation.

We show how a simple baseline outperforms all existing methods on current and proposed benchmarks, without even being able to accumulate knowledge over long periods of time.

References

[1] Abnar, S., Berg, R. v. d., Ghiasi, G., Dehghani, M., Kalchbrenner, N., and Sedghi, H. (2021). Gradual domain adaptation in the wild: When intermediate distributions are absent. *arXiv preprint arXiv:2106.06080*.

[2] Bobu, A., Tzeng, E., Hoffman, J., and Darrell, T. (2018). Adapting to continuously shifting domains. *Workshop Track - ICLR 2018*.

[3] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

[4] Feng, F., Chan, R. H., Shi, X., Zhang, Y., and She, Q. (2019). Challenges in task incremental learning for assistive robotics. *IEEE Access*, 8:3434–3441.

[5] Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., and Lempitsky, V. (2016). Domain-adversarial training of neural networks. *The journal of machine learning research*, 17(1):2096–2030.

[6] Geirhos, R., Rubisch, P., Michaelis, C., Bethge, M., Wichmann, F. A., and Brendel, W. (2018). Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. *arXiv preprint arXiv:1811.12231*.

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

[8] Goyal, S., Sun, M., Raghunathan, A., and Kolter, Z. (2022). Test-time adaptation via conjugate pseudo-labels. *arXiv preprint arXiv:2207.09640*.

[9] Han, J., Liang, X., Xu, H., Chen, K., Lanqing, H., Mao, J., Ye, C., Zhang, W., Li, Z., Liang, X., et al. (2021). Soda10m: A large-scale 2d self/semi-supervised object detection dataset for autonomous driving. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

[10] He, K., Zhang, X., Ren, S., and Sun, J. (2016a). Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778. IEEE Computer Society.

[11] He, K., Zhang, X., Ren, S., and Sun, J. (2016b). Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778. IEEE Computer Society.

[12] Hendrycks, D., Basart, S., Mu, N., Kadavath, S., Wang, F., Dorundo, E., Desai, R., Zhu, T., Parajuli, S., Guo, M., et al. (2020a). The many faces of robustness: A critical analysis of out-of-distribution generalization. *ArXiv preprint*, abs/2006.16241.

[13] Hendrycks, D. and Dietterich, T. (2019a). Benchmarking neural network robustness to common corruptions and perturbations. *arXiv preprint arXiv:1903.12261*.

[14] Hendrycks, D. and Dietterich, T. G. (2019b). Benchmarking neural network robustness to common corruptions and perturbations. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

[15] Hendrycks, D., Mu, N., Cubuk, E. D., Zoph, B., Gilmer, J., and Lakshminarayanan, B. (2020b). Augmix: A simple data processing method to improve robustness and uncertainty. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

[16] Hoffman, J., Darrell, T., and Saenko, K. (2014). Continuous manifold based adaptation for evolving visual domains. In *Computer Vision and Pattern Recognition (CVPR)*.

[17] Kar, O. F., Yeo, T., Atanov, A., and Zamir, A. (2022). 3d common corruptions and data augmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18963–18974.

[18] Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al. (2017). Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526.

[19] Lee, D.-H. (2013). Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *ICML Workshop : Challenges in Representation Learning (WREPL)*.

[20] Liang, J., Hu, D., Wang, Y., He, R., and Feng, J. (2021). Source data-absent unsupervised domain adaptation through hypothesis transfer and labeling transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

[21] Liu, Z., Miao, Z., Pan, X., Zhan, X., Lin, D., Yu, S. X., and Gong, B. (2020). Open compound domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12406–12415.

[22] Lomonaco, V. and Maltoni, D. (2017). Core50: a new dataset and benchmark for continuous object recognition. In Levine, S., Vanhoucke, V., and Goldberg, K., editors, *Proceedings of the 1st Annual Conference on Robot Learning*, volume 78 of *Proceedings of Machine Learning Research*, pages 17–26. PMLR.

[23] Mahajan, D., Girshick, R., Ramanathan, V., He, K., Paluri, M., Li, Y., Bharambe, A., and van der Maaten, L. (2018). Exploring the limits of weakly supervised pretraining. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

[24] Motiian, S., Jones, Q., Iranmanesh, S., and Doretto, G. (2017). Few-shot adversarial domain adaptation. *Advances in neural information processing systems*, 30.

[25] Mummadi, C. K., Hutmacher, R., Rambach, K., Levinkov, E., Brox, T., and Metzen, J. H. (2021). Test-time adaptation to distribution shift by confidence maximization and input transformation. *arXiv preprint arXiv:2106.14999*.

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

[27] Niu, S., Wu, J., Zhang, Y., Chen, Y., Zheng, S., Zhao, P., and Tan, M. (2022). Efficient test-time model adaptation without forgetting. *arXiv preprint arXiv:2204.02610*.

[28] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. (2019). Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

[29] Prabhu, A., Torr, P. H., and Dokania, P. K. (2020). Gdumb: A simple approach that questions our progress in continual learning. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, pages 524–540. Springer.

[30] Press, O., Schneider, S., Kuemmerer, M., and Bethge, M. (2022). Ccc: Continuously changing corruptions. In *ICML 2022 Shift Happens Workshop*.

[31] Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. (2021). Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR.

[32] Ratcliff, R. (1990). Connectionist models of recognition memory: constraints imposed by learning and forgetting functions. *Psychological review*, 97(2):285.

[33] Rusak, E., Schneider, S., Pachitariu, G., Eck, L., Gehler, P. V., Bringmann, O., Brendel, W., and Bethge, M. (2021). If your data distribution shifts, use self-learning. *Transactions of Machine Learning Research*.

[34] Rusak, E., Schott, L., Zimmermann, R., Bitterwolf, J., Bringmann, O., Bethge, M., and Brendel, W. (2020). Increasing the robustness of dnns against image corruptions by playing the game of noise. *ArXiv preprint*, abs/2001.06057.

[35] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. (2015). Imagenet large scale visual recognition challenge. *International journal of computer vision (IJCV)*.

[36] Schneider, S., Rusak, E., Eck, L., Bringmann, O., Brendel, W., and Bethge, M. (2020). Improving robustness against common corruptions by covariate shift adaptation. In *Advances in neural information processing systems*.

[37] Shi, X., Li, D., Zhao, P., Tian, Q., Tian, Y., Long, Q., Zhu, C., Song, J., Qiao, F., Song, L., Guo, Y., Wang, Z., Zhang, Y., Qin, B., Yang, W., Wang, F., Chan, R. H. M., and She, Q. (2020). Are we ready for service robots? the OpenLORIS-Scene datasets for lifelong SLAM. In *2020 International Conference on Robotics and Automation (ICRA)*, pages 3139–3145.

[38] Sun, P., Kretzschmar, H., Dotiwalla, X., Chouard, A., Patnaik, V., Tsui, P., Guo, J., Zhou, Y., Chai, Y., Caine, B., Vasudevan, V., Han, W., Ngiam, J., Zhao, H., Timofeev, A., Ettinger, S., Krivokon, M., Gao, A., Joshi, A., Zhang, Y., Shlens, J., Chen, Z., and Anguelov, D. (2020). Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

[39] Sun, Y., Tzeng, E., Darrell, T., and Efros, A. A. (2019a). Unsupervised domain adaptation through self-supervision. *ArXiv preprint*, abs/1909.11825.

[40] Sun, Y., Wang, X., Liu, Z., Miller, J., Efros, A. A., and Hardt, M. (2019b). Test-time training for out-of-distribution generalization. *ArXiv preprint*, abs/1909.13231.

[41] Tzeng, E., Hoffman, J., Saenko, K., and Darrell, T. (2017). Adversarial discriminative domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7167–7176.

[42] Wang, D., Shelhamer, E., Liu, S., Olshausen, B., and Darrell, T. (2020a). Fully test-time adaptation by entropy minimization. *ArXiv preprint*, abs/2006.10726.

[43] Wang, D., Shelhamer, E., Liu, S., Olshausen, B., and Darrell, T. (2020b). Tent: Fully test-time adaptation by entropy minimization. *arXiv preprint arXiv:2006.10726*.

[44] Wang, Q., Fink, O., Van Gool, L., and Dai, D. (2022). Continual test-time domain adaptation. *arXiv preprint arXiv:2203.13591*.

[45] Wulfmeier, M., Bewley, A., and Posner, I. (2018). Incremental adversarial domain adaptation for continually changing environments. In *2018 IEEE International conference on robotics and automation (ICRA)*, pages 4489–4495. IEEE.

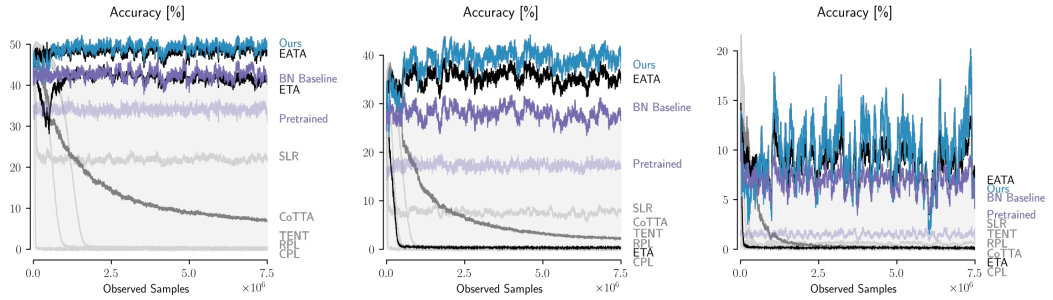
[46] Xie, Q., Luong, M., Hovy, E. H., and Le, Q. V. (2020). Self-training with noisy student improves imagenet classification. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 10684–10695. IEEE.

[47] Yu, F., Chen, H., Wang, X., Xian, W., Chen, Y., Liu, F., Madhavan, V., and Darrell, T. (2020). Bdd100k: A diverse driving dataset for heterogeneous multitask learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

[48] Yue, X., Zheng, Z., Zhang, S., Gao, Y., Darrell, T., Keutzer, K., and Vincentelli, A. S. (2021). Prototypical cross-domain self-supervised learning for few-shot unsupervised domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13834–13844.

[49] Zhang, M., Levine, S., and Finn, C. (2022). Memo: Test time robustness via adaptation and augmentation. *Advances in Neural Information Processing Systems*, 35:38629–38642.

A CCC Plots



(a) (b) (c)

Figure 7: Adaptation performance of all evaluated models depending on the number of observed samples so far. (a) CCC Easy. (b) CCC Medium. (c) CCC Hard. For all subplots, model performances are averaged over the 9 runs of the respective difficulty level.

B Path Finding Algorithm

Algorithm 1 describes the pseudo code of the algorithm used to generate CCC. The algorithm is based on a set of Calibration Matrices, similar to the one shown in Figure 3. There exists a matrix m for every (n_a, n_b) pair, such that $m[i][j]$ is equal to accuracy of a pretrained ResNet-50 on the comination of noises (n_a, n_b), and severities $(i/5, j/5)$. We will release the full set of matrices upon publication.

Additionally, algorithm 2 uses a function *MinValidPath*(s_1, s_2): this function returns the minimum path that starts at (s_1, s_2) and ends at $(0, s_j)$ for some s_k. The cost of a path is simply the average of all entries along the path. The minimum path is defined as the path with a cost closest to b_a in absolute terms. Lastly, a path is only valid if it starts with s_2 equal to 0, every transition either decreases s_1 by 0.25, or increases s_2 by 0.25, and stops once s_1 is equal to 0.

Algorithm 1 Algorithm used to generate each split of CCC

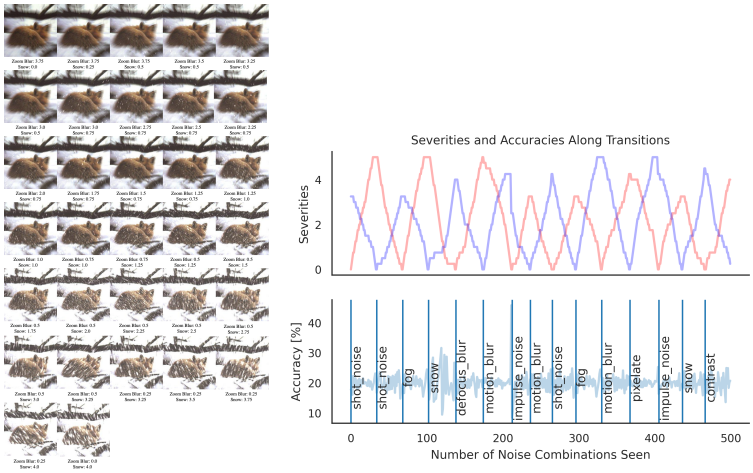
Require: ba, k, T ▷ Baseline accuracy, transition speed, and total split size.
$t = 0$ ▷ Initialize the total images generated counter
$n_1, n_2 \sim \text{Uniform}(\{1 \dots 15\})$ ▷ Initialize the first two corruptions.
path $\leftarrow \text{CalculatePath}(n_1, n_2, ba)$ ▷ Calculate path along the noise pair with an average accuracy closest to ba.
loop
  $s_1, s_2 \leftarrow \text{path}[p]$
  Subset $\sim \text{Uniform}(\text{ImageNetVal})$
  apply (n_1, s_1, n_2, s_2) to Subset
  save Subset
  $t \leftarrow k$
  if $t >= T$ **then** return
  end if
  if $p = \text{len}(\text{path}) - 1$ **then**
    $n_1 \leftarrow n_2$
    $n_2 \sim \text{Uniform}(\{1 \dots 14\})$
    path $\leftarrow \text{CalculatePath}(n_1, n_2, ba)$ ▷ Calculate new path along the new noise pair.
    $p \leftarrow 0$
  else
    $p \leftarrow p + 1$ ▷ Move to the next severity combination
  end if
end loop

Algorithm 2 CalculatePath

Require: n_1, n_2, ba ▷ The 2 noises, and the baseline accuracy

$m \leftarrow \text{CalibrationMatrix}[n_1][n_2]$
$\text{MinPath}, \text{MinCost} \leftarrow \text{MinValidPath}(0, 0, m, ba)$
for $s_1 \in 0.25, 0.5, 0.75, \ldots, 5$ **do**
    $m \leftarrow \text{MinValidPath}(1, 0, m, ba)$
    $\text{MinPath}, \text{MinCost} \leftarrow \text{MinValidPath}(s_1, 0, m, ba)$
end for
return $MinPath$

The resulting dataset features transitions between two different noises like the ones seen in Figure 8 (a). We additionally plot the accuracy of a pretrained ResNet-50, alongisde the severities of the different noises in Figure 8 (b).



(a) (b)

Figure 8: **(a)** Visualization of CCC-Medium's smooth transition between Zoom Blur to Snow. Note: CCC additionally uses random flips and crops, which are not shown here. **(b)** As CCC transitions between noises, the severities of the first noise (red), and the second noise (blue) go up and down correspondingly, in order to keep the accuracy of a pretrained model stable.

C Episodic ETA

The objective uses an entropy loss $E(x;\Theta)$ which is weighted for each sample through $S(.)$ [18]:

$$S(x)E(x;\Theta) = -S(x)\sum_{y\in C} f_\Theta(y|x)\log f_\Theta(y|x) \tag{1}$$

$S(.)$ is built from two components, namely: S^{ent} that filters out samples that have an entropy higher than E_0, and weighs the remaining samples according to their entropy.

$$S^{ent} = \frac{1}{\exp[E(x;\Theta) - E_0]}\mathbb{1}_{E(x;\Theta)<E_0}(x) \tag{2}$$

The second component is a redundancy filter that filters out samples that are too similar to previously seen inputs.

$$S^{div}(x) = \mathbb{1}_{cos(f_\Theta, m^{t-1})<\epsilon}(x) \tag{3}$$

Where:

$$m^t = \begin{cases} \bar{y}_1 & \text{if } t = 1 \\ \alpha\bar{y}_t + (1-\alpha)m^{t-1} & \text{if } t > 1 \end{cases} \tag{4}$$

Where $\bar{y} = \frac{1}{n}\sum_{k=1}^{n}\hat{y}_k^t$ is the model's average prediction at step t, averaged over a batch of n samples. We use the original values of E_0, α, and ϵ. We show that our method is less sensitive to E_0 in section 5. Finally, $S(.)$ is given as:

$$S(x) = S^{ent}(x)\cdot S^{div}(x) \tag{5}$$

D EATA Implementation and Ablations

Our implementation of EATA differs from the official implementation. The reason for this is that the official implementation uses clean ImageNet validation images to calculate the Fisher vector matrix for its regularizer (line 44). This stands in contradiction with the method, which should not have access to the training distribution at test time, as shown in the paper in Table 1.

Instead of using $2,000$ ImageNet validation images, we calculate the Fisher matrix using the first $2,000$ images in our data stream. We conduct a hyperparameter search on the weight regularizer tradeoff parameter β:

β	25	50	100	250	k00	1000	1500	2000
Acc. [%]	46.5	46.9	**47.1**	46.7	46.1	45.6	44.8	44.0

Table 5: Accuracy of EATA on CIN-C holdout noises for different values of the weight regularizer loss.

Using the optimal value, 100 led to worse results than the default value, 2000, on CCC:

	CIN-C	CCC-Easy	CCC-Medium	CCC-Hard	CCC Avg
EATA-100	46.7	47.7	36.5	3.8	29.3
EATA-2000	41.8	48.2	35.4	8.7	30.8
Ours	46.5	49.3	38.9	9.6	32.6

Table 6: Accuracy of EATA on CIN-C holdout noises for different values of the Fisher alpha.

In the end, we used the original value of 2000, as that was optimal on the CCC dataset.

In addition, we conducted a hyperparameter search for EATA on a ViT backbone. As shown in 6, EATA performs worse than a pretrained, non adapting baseline in this setting. As with the previous experiment, the original value of 2000 is optimal.

β	1000	2000	3000	4000
Acc. [%]	34.7	**37.7**	27.0	16.3

Table 7: Accuracy of EATA on CCC-Medium using a ViT backbone, for different values of the regularizer, β.

E Novelty of Resetting

Our work is the first to propose resetting to solve collapse in TTA methods. Notably, while prior work [27, 43, 49] has briefly touched upon the concept of episodic resetting, the methodology and its application is significantly distinct and unrelated to collapse in TTA.

- **Tent** [43] mentions episodic in the context of overfitting to a single sample in segmentation (similar to [49]'s overfitting to a single sample). Resetting here is unrelated to collapse, as the paper doesn't discuss collapse at all.
- Although **MEMO** [49] uses resetting, it does so because it overfits to one image (and its augmentations) every step. MEMO doesn't discuss collapse or catastrophic forgetting. MEMO compares itself to a version of Tent that resets after every step (which they call Tent + episodic resetting), because MEMO without augmentations is similar to Tent + episodic resetting with a batch size of 1. (Note: MEMO is outperformed by BN/Tent/ETA when using the standard batch size of 64)
- **EATA** [27] shows results for Tent + episodic resetting after every step in its tables, but provides no reasoning or discussion for doing this. Tent + episodic resetting is outperformed by regular Tent.

F Compute details

We conduct all experiments on Nvidia RTX 2080 TI GPUs with 12GB memory per device. All experiments except our study on larger models were conducted on a single GPU. For CoTTA experiments, we use data parallel training on 2 GPUs. A bulk of the compute spent for this work was on computing baseline accuracies on the calibration dataset, which contains 463M images.

G Software and Dataset Licenses

G.1 Datasets

- ImageNet-C [13]: Creative Commons Attribution 4.0 International, `https://zenodo.org/record/2235448`
- ImageNet-C [13], code for generating corruptions `https://github.com/hendrycks/robustness`
- ImageNet-3D-CC [17]: CC-BY-NC 4.0 License `https://github.com/EPFL-VILAB/3DCommonCorruptions`

G.2 Models

- PyTorch's [28] ResNet [11] `https://pytorch.org/vision/stable/models.html`
- Adaptive BN [26, 36]: Apache License 2.0, `https://github.com/bethgelab/robustness`
- TENT [43]: MIT License, `https://github.com/DequanWang/tent`
- RPL [33]: Apache License 2.0, `https://github.com/bethgelab/robustness`
- CoTTA [44]: MIT License, `https://github.com/qinenergy/cotta`
- CPL [8]: `https://github.com/locuslab/tta_conjugate`
- EATA [27]: `https://github.com/mr-eggplant/EATA`