This WACV paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

Layer-wise Auto-Weighting for Non-Stationary Test-Time Adaptation

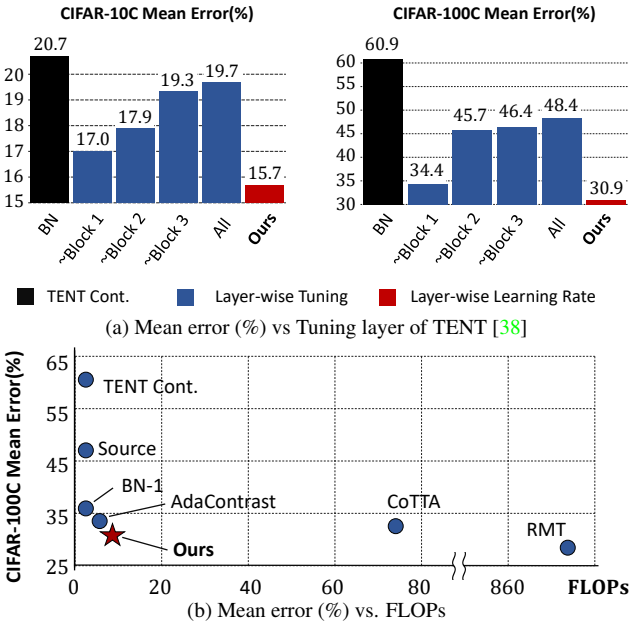
Junyoung Park[1] Jin Kim[1] Hyeongjun Kwon[1] Ilhoon Yoon[1] Kwanghoon Sohn[1,2*]
[1]Yonsei University [2]Korea Institute of Science and Technology (KIST)
{jun_yonsei, kimjin928, kwonjunn01, ilhoon231, khsohn}@yonsei.ac.kr

Abstract

Given the inevitability of domain shifts during inference in real-world applications, test-time adaptation (TTA) is essential for model adaptation after deployment. However, the real-world scenario of continuously changing target distributions presents challenges including catastrophic forgetting and error accumulation. Existing TTA methods for non-stationary domain shifts, while effective, incur excessive computational load, making them impractical for on-device settings. In this paper, we introduce a layer-wise auto-weighting algorithm for continual and gradual TTA that autonomously identifies layers for preservation or concentrated adaptation. By leveraging the Fisher Information Matrix (FIM), we first design the learning weight to selectively focus on layers associated with log-likelihood changes while preserving unrelated ones. Then, we further propose an exponential min-max scaler to make certain layers nearly frozen while mitigating outliers. This minimizes forgetting and error accumulation, leading to efficient adaptation to non-stationary target distribution. Experiments on CIFAR-10C, CIFAR-100C, and ImageNet-C show our method outperforms conventional continual and gradual TTA approaches while significantly reducing computational load, highlighting the importance of FIM-based learning weight in adapting to continuously or gradually shifting target domains. [1]

1. Introduction

Despite the recent advances in deep learning [9, 15–17], deep neural networks (DNNs) frequently encounter performance degradation when the source and target domains are different [19,27,40]. For instance, a pre-trained classification model suffers from this phenomenon when tested on corrupted images due to sensor deterioration. Among various efforts to address these domain shifts, *test-time adaptation* (TTA) has recently received significant attention, especially for its practicality [8, 38]. TTA aims to adapt the pre-trained source model by learning from unlabeled target data during inference where access to the source data is no longer available. Since source data is unavailable during inference due to privacy concerns or legal constraints, TTA poses a realistic problem and is more challenging than unsupervised domain adaptation. TTA can also be conducted in online scenarios where revisiting the past test samples is prohibited, adapting the model instantly



(a) Mean error (%) vs Tuning layer of TENT [38]

(b) Mean error (%) vs. FLOPs

Figure 1. **(a) Continual TTA Performance between gradual layer tuning and Ours** in CIFAR-C dataset. While heuristically selecting layers to tune achieves favorable performance, our method outperforms through layer-wise auto-weighting. **(b) Adaptation performance of CTTA methods** on CIFAR-100C benchmarks. The $x-$ and $y-$axis are the FLoating point Operations Per second (FLOPs) and mean error (%). Our method outperforms others with a significantly less computational load.

*Corresponding author

[1]Code is available at https://github.com/junia3/LayerwiseTTA

This research was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (NRF2021R1A2C2006703), and partly supported by the Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No.2021-0-02068, Artificial Intelligence Innovation Hub).

using only the current test batch. This makes TTA more applicable in on-device settings [11, 26, 40]

Existing TTA methods often tackle the domain shift between the source and a fixed target domain by using pseudo labels or entropy regularization [33, 38]. While these self-training methods have shown effectiveness, their improvements are demonstrated only in a single domain shift scenario at a time. Since real-world target distribution tends to change continuously, it is necessary to consider the non-stationary target domain. [39] introduced *continual test-time adaptation* (CTTA) where the model is adapted to a sequence of domain shifts. The two challenges of CTTA are error accumulation by miscalibrated pseudo labels [13] and catastrophic forgetting by the gradual dilution of knowledge from the source pre-trained model. To prevent the error accumulation, pioneering attempts [8, 39] employ augmentation-averaged pseudo labels and an exponential moving averaged (EMA) model. For source knowledge preservation, random parameter restore [39] and source prototypes [8] are used. However, using these additional processes requires an excessive computational load than the original model as shown in Figure 1b. This makes the methods less practical to use in on-device settings.

Despite the advancements in CTTA, previous works have not considered the heterogeneity of layers: each layer has a distinct role and captures different information [2, 12, 29, 43]. Recently, in the context of transfer learning, [23] identified certain layers of a pre-trained model that are already near-optimal for the target data through a heuristic approach. They demonstrated that optimizing only the remaining layers helps preserve useful information from pre-training while enabling efficient learning for the target distribution. In our preliminary CTTA experiment shown in Figure 1a, tuning the subset of layers improved adaptation capability compared to the vanilla or standard fine-tuning of the entropy minimization [38]. We hypothesize that there is potential for further improvement of adaptation to continuously shifting target domains by identifying an optimal combination of tuning layers.

Inspired by this observation, we propose a novel layer-wise auto-weighting algorithm that autonomously identifies the layers that need preservation or concentrated adaptation. The goal of this approach is to utilize the knowledge gained during source pre-training and efficiently adapt to the non-stationary target distribution. To achieve this, we employ the Fisher Information Matrix (FIM) to approximate the second derivative of the log-likelihood [32]. We calculate the FIM-based learning weight of each layer to measure the sharpness of the log-likelihood-layer parameters surface on target data [7]. Since higher sharpness indicates parameter sensitivity, we can identify layers to update or preserve. Moreover, we introduce an exponential min-max scaler to amplify the learning weight difference across layers. It makes certain layers nearly frozen while mitigating distortion of learning weights outliers. As a result, our method can selectively focus on layers associated with log-likelihood changes in the target domain while preserving the unrelated ones. Our method outperforms conventional CTTA approaches while significantly reducing computational load. Experiments and ablation studies conducted on various benchmarks and networks, such as CIFAR-10C, CIFAR-100C, and ImageNet-C, prove the significance of the FIM-based learning weight. Our key contributions are as follows:

- We propose the layer-wise auto-weighted learning algorithm by leveraging the Fisher Information Matrix (FIM) to autonomously identify layers needing preservation or concentrated adaptation.
- We introduce an exponential min-max scaler to amplify the learning weight difference while mitigating distortion from outliers.
- We demonstrate comparable performance of our method while reducing computational load, compared to conventional approaches in both continual and gradual TTA on various benchmarks and networks.

2. Related Work

Test-time adaptation. The goal of test-time adaptation (TTA) is to adapt source pre-trained models to the unlabeled target domain with no access to the source dataset. Moreover, TTA can be considered an online manner where the predictions are needed immediately and the model is adapted using only the current test batch. It has recently gained increasing attention [1, 4, 24, 26, 33, 38, 44]. The pioneering work, BN-1 [33] aligns the Batch Normalization (BN) statistics to the target domain while TENT [38] minimizes Shannon entropy [35] optimizing only the BN parameters. Instead of updating only a portion of the network, Adacontrast [3] leverages the memory module with contrastive learning and updates the entire network. Although previous TTA methods show significant improvement, they are only demonstrated for a single domain shift at a time.

Non-stationary Test-time Adaptation. Typically, TTA operates under the assumption of a stationary scenario without specific constraints, but the real-world domain shifts are dynamic and constantly changing. In this scenario, the non-stationary TTA settings can be categorized into two branches: Continual Test-Time Adaptation (CTTA) and Gradual Test-Time Adaptation (GTTA). CoTTA [39], the pioneering work of CTTA, was proposed to adapt continual domain shift in test-time.

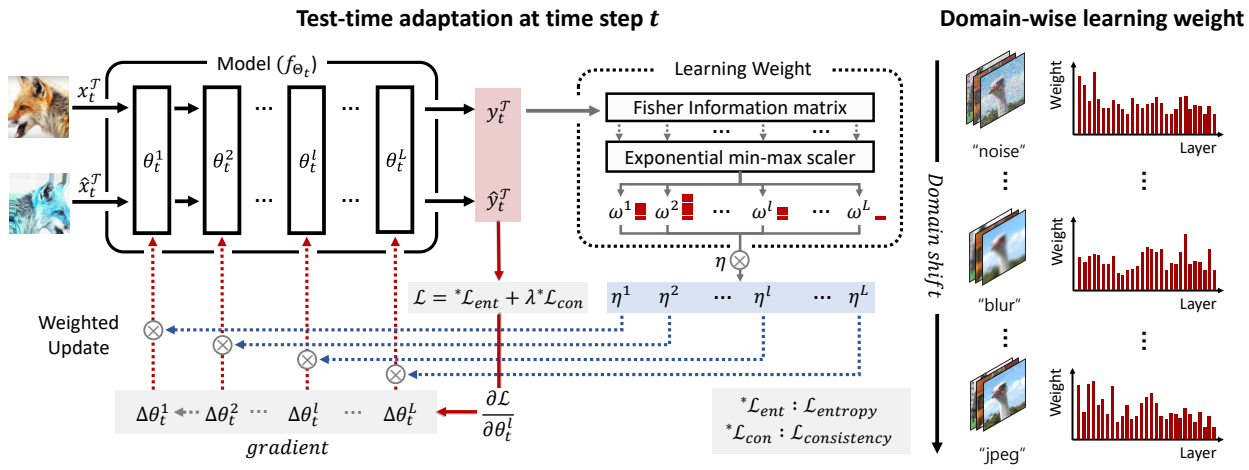


Figure 2. **Layer-wise learning rate framework:** The model is adapted to current test batch $x_t^{\mathcal{T}}$. Before updating the current model parameter, the Fisher Information Matrix (FIM) is empirically obtained from log-likelihood w.r.t. current batch images and update domain-level FIM. We further employ an exponential min-max scaler to mitigate the distortion of learning weights from outliers. Finally, we can automatically update with adjusted learning weight and total objective $\mathcal{L}$ in Eq. (9)

They employ an exponential moving average (EMA) model to prevent error accumulation from miscalibration [13] and catastrophic forgetting. In addition, RMT [8] leverages source prototypes for indirectly aligning the source distribution. Moreover, gradual TTA [28] assumes a gradual domain change in severity rather than an abrupt shift. GTTA [28] employs style transfer to fill the gap between intermediate distribution shifts. However, they still suffer from heavy computational loads since using the auxiliary part like the EMA models.

Layer-wise fine-tuning. In transfer learning, preserving information from the source pre-training is important. There are prominent approaches, such as partial parameter freezing [10, 14, 22, 25, 30, 31], and leveraging layer criticality refer to the loss surface [2, 29, 43]. Recently, surgical fine-tuning [23] was proposed to heuristically find already near-optimal layers to the target domain and fine-tune only the remaining subset of all layers in the pre-trained models. However, previous works addressed the case that the ground truth of target data is available. In this paper, we propose a layer-wise auto-weighted adaptation method applicable to the unlabeled target data.

Fisher information matrix. The Fisher information matrix (FIM) is a fundamental concept for defining intrinsic structures and performing gradient-based optimization. From this perspective, Tan *et al.* [37] claims that FIM can achieve better results due to the adaptation of curvature information since it is a kind of second-order derivative. Therefore, by conducting FIM in DNNs, FIM could adaptively adjust the step length throughout the training process. In this regard, several works [20, 34] employ FIM to contiual learning to prevent catastrophic forgetting via adjusting the learning rate. Despite the advantage of FIM, test-time adaptation with FIM, to the best of our knowledge, has not been demonstrated so far. In this paper, we propose a simple but effective approach to adjust the learning rate by measuring the distribution shift via FIM.

3. Proposed Method

In real-world applications, the environment dynamically changes. To effectively adapt to non-stationary test data, we propose an algorithm that autonomously assigns weights to layers needing preservation or concentrated adaptation. Therefore, our method can adapt efficiently by preserving the source knowledge. In this section, We revisit the problem statement of TTA in Sec. 3.1 and introduce the preliminaries and layer-wise weighted learning rate in Sec. 3.2. We also introduce the exponential min-max scaler that enables stable adaptation with main task loss in Sec. 3.3. Finally, we show our overall framework with consistency loss in Sec. 3.4. The overall procedure of our method is illustrated in Figure 2.

3.1. Problem statement

The goal of continual and gradual test time adaptation is to adapt the pre-trained model to a domain that is consistently changing in an online manner without access to the source dataset. Given the model f_{Θ_0} where the pre-trained parameter Θ_0 trained on the source domain $\mathcal{D}^{\mathcal{S}} = \{\mathcal{X}^{\mathcal{S}}, \mathcal{Y}^{\mathcal{S}}\}$, our goal is to adapt the model to the unlabeled target domain $\mathcal{D}^{\mathcal{T}} = \{\mathcal{X}^{\mathcal{T}}\}$, which consists of multiple domains that change sequentially. At each time step t, the model f_{Θ_t} adapts to the target domain samples $x_t^{\mathcal{T}} \in \mathcal{X}^{\mathcal{T}}$, resulting in the updated model $f_{\Theta_{t+1}}$. The model parameter Θ_t is composed of parameters θ_t^l for each layer l, such that $\Theta_t = \{\theta_t^1, \cdots, \theta_t^l, \cdots, \theta_t^L\}$. Given the TTA loss function

$\mathcal{L}$, the optimization of θ_t^l is as follows:

$$\Delta\theta_t^l = \frac{\partial \mathcal{L}(x_t^{\mathcal{T}}; \Theta_t)}{\partial \theta_t^l}, \quad \theta_{t+1}^l := \theta_t^l - \eta * \Delta\theta_t^l. \tag{1}$$

where η denotes the learning rate, which is typically set to be consistent across all layers.

3.2. Layer-wise Learning Weight

Non-stationary domain adaptation aims to not only optimize effectively the target domain but also prevent error accumulation. To achieve these objectives, we present the layer-wise learning weight to automatically adjust the learning rate η depending on models and domain changes. First, we have to infer the sharpness of the log-likelihood and the layer parameter θ_t^l surface by calculating the second derivative of the log-likelihood [7]. We can get the value of second derivatives through the Hessian matrix of the layer parameter θ_t^l. However, a direct computation of the Hessian matrix often fails on some large models [17, 41, 42] since the Hessian matrix increases quadratically $\mathbb{R}^{|\theta_t^l|}$ with the number of model parameters $|\theta_t^l|$ [37].

Approximation of the Hessian matrix. To mitigate computational overflow, we employ the Fisher information matrix (FIM). The Hessian matrix requires two steps of derivatives. However, the FIM can approximate the Hessian matrix of log-likelihood with one step of gradient [32]. This distinction can help reduce the aforementioned problem, computational load. To get FIM, we calculate log-likelihood from predicted output $p_{\Theta_t}(x_t^{\mathcal{T}})$ of the current test batch $x_t^{\mathcal{T}}$. Then we can get the layer-wise score function $s(\theta_t^l; \cdot)$ which is the gradient of layer parameter θ_t^l from the log-likelihood as follows:

$$s(\theta_t^l; x_t^{\mathcal{T}}) = \nabla_{\theta_t^l} \log\left(p_{\Theta_t}(x_t^{\mathcal{T}})\right) \tag{2}$$

Then, the layer-wise FIM can be formulated as follows:

$$I_t^l = \mathbb{E}_{x_t^{\mathcal{T}}}\left[s(\theta_t^l; x_t^{\mathcal{T}}) s(\theta_t^l; x_t^{\mathcal{T}})^\top\right]. \tag{3}$$

Note that the layer-wise FIM I_t^l is calculated for each layer θ_t^l with respect to the current test batch $x_t^{\mathcal{T}}$. In our implementation, the layer-wise FIM is calculated based on the gradient of the layer computed using the negative log likelihood loss. We further propose domain-level FIM $\tilde{I}_t^l$ from layer-wise FIM I_t^l at time step t, inspired by [34]. The domain-level FIM is formulated as:

$$\tilde{I}_t^l = \tilde{I}_{t-1}^l + I_t^l. \tag{4}$$

Since the domain-level FIM $\tilde{I}_t^l$ is derived by accumulation rather than solely utilizing the current FIM, it includes more information about domain and model characteristics. With the domain-level FIM, layer-wise learning weight can be obtained by taking a trace of FIM as follows:

$$w^l = \sqrt{\mathrm{Tr}(\tilde{I}_t^l)}. \tag{5}$$

Layer-wise learning rate. Based on the domain-level FIM in Eq. (4), we computed the learning weight w^l for each layer. The simplest way of layer-wise weighting is as follows:

$$\eta^l = \eta * w^l. \tag{6}$$

Using this sharpness-based [7] auto-weighting framework, the model efficiently adapts to the varying target domain by distributing layer-wise learning rates η^l.

3.3. Exponential min-max scaler

However, the calculated learning weights are unbounded, which can deteriorate stable adaptation. To realize this, we design an exponential min-max scaler to bound learning weights within the range of 0 and 1. The exponential min-max scaler is employed by as follows:

$$\eta^l = \eta * \bar{w}^l, \;\; \bar{w}^l = \left(\frac{w^l - w_{\min}^l}{w_{\max}^l - w_{\min}^l + \epsilon}\right)^\tau, \tag{7}$$

where τ is the exponential hyperparameter to mitigate the adverse effect of the outlier. We set $\tau = 1$ when the learning weights are scaled in a proper way. With this scaler, we can amplify the learning weight difference across layers making certain layers nearly frozen. Moreover, it is more robust about the outlier value of the FIM than the vanilla min-max normalization. Because when τ is greater than 1, it reduces the over-scaled learning weights due to an excessively small minimum. Conversely, as τ is lower than 1, the under-scaled learning weights owing to a relatively large maximum can be boosted. We can also mitigate the distortion of learning weights from outliers resulting in stable adaptation.

3.4. Overall Update

Consistency Loss. Although layer-wise auto-weighted learning via FIM can reduce error accumulation on differential updates, there still exists the miscalibrated problem. This miscalibration represents the difference between the information inherent in the data and the information the network acquires, due to inaccessibility to labels. To address this problem, inspired from [3, 36], we employ a self-training scheme in which the prediction of the original input batch $x_t^{\mathcal{T}}$ is considered as the pseudo label for the augmented batch $\hat{x}_t^{\mathcal{T}}$. We further propose consistency loss $\mathcal{L}_{\text{consistency}}$ to regularize the network and prevent degradation from miscalibrated information. The

consistency loss $\mathcal{L}_{\text{consistency}}$ is formulated via conventional cross-entropy loss such that:

$$\mathcal{L}_{\text{consistency}} = -\mathbb{E}_{x_t^{\mathcal{T}}} \left[\sum_{c=1}^{C} \sigma(y_t^{\mathcal{T}}) \log \sigma(\hat{y}_t^{\mathcal{T}}) \right], \quad (8)$$

Where $y_t^{\mathcal{T}}$, $\hat{y}_t^{\mathcal{T}}$ are the corresponding logits and $\sigma(\cdot)$ is the sigmoid function.

Loss function. Building upon negative log-likelihood, we compute the Fisher Information Matrix (FIM) and determine layer-wise weighted learning rates η^l. Optimization operates batch-wise with derived learning rates. Our overall loss function is composed of Shannon entropy loss $\mathcal{L}_{\text{entropy}}$ from the TENT [38] and consistency loss such that:

$$\mathcal{L} = \mathcal{L}_{\text{entropy}} + \lambda \mathcal{L}_{\text{consistency}}, \quad (9)$$

where $\mathcal{L}_{\text{entropy}}$ and $\mathcal{L}_{\text{consistency}}$ are the sum of each term for all samples in a batch, and the λ controls the stability originated from the number of the class.

Weighted Update. Our full optimization framework is briefly visualized with Figure 2 and Algorithm 1. The layer-wise auto-weighted learning procedure is summarized as follows: (1) In the scenario of non-stationary test-time adaptation, the parameters of the previously trained network remain unchanged for the subsequent optimization phases. The log-likelihood is computed using the predicted logits based on the prior of the previous model's parameters. (2) Gradient computation is performed for each layer based on the log-likelihood and using the computed gradients, the empirical FIM is approximated as an estimation of the Hessian. FIM is calculated online as a domain-level FIM according to Eq. (4). (3) The layer importance, determined by characterizing the trace of the FIM, is used to adaptively adjust the learning rate for each parameter. (4) According to the previously mentioned exponential min-max scaler Eq. (7), we updated each layer of our model with the gradient $\Delta\theta_t^l$ with respect to the loss function $\mathcal{L}$ in a layer-wise manner, as follows:

$$\theta_{t+1}^l := \theta_t^l - \eta^l * \Delta\theta_t^l. \quad (10)$$

This allows for the separation of the importance of each learnable parameter embedded in each layer, enabling optimization with different step sizes.

Algorithm 1: Layer-wise auto-weighed learning

Require: The encoder f, pretrained parameter Θ_0, unlabeled target domain test dataset $\mathcal{X}^{\mathcal{T}}$

for *each time step* t **do**
- Draw an i.i.d. batch $x_t^{\mathcal{T}} \in \mathcal{X}^{\mathcal{T}}$,
- Compute score function $s(\theta_t^l; x_t^{\mathcal{T}})$ according to Eq. (2),
- Compute layer-wise Fisher Information Matrix I_t^l per layer according to Eq. (3),
- Update domain-level FIM according to Eq. (4),
- Compute the learning weight of each layer using a trace of domain-level FIM according to Eq. (5),
- Apply exponential min-max scaler to learning weight w^l according to Eq. (7),
- Update learning rate according to $\eta^l \leftarrow \eta * \bar{w}^l$,
- Calculate gradient with objective function $\mathcal{L}$ in Eq. (9),
- Update Θ_t to Θ_{t+1} by gradient descent with layer-wise learning rate in Eq. (10)

4. Experiments

We present the experimental results to introduce the effectiveness and efficiency of our proposed method. We compare our method with state-of-the-art methods in different settings of non-stationary test-time adaptation: continual test-time adaptation (CTTA) and gradual test-time adaptation (GTTA), which results are presented Sec. 4.2, 4.3 respectively. Then, we provide the results of extensive ablation studies and analysis in Sec. 4.4.

4.1. Experimental Settings

Datasets and metrics. We evaluate the non-stationary test-time adaptation of our proposed method on CIFAR-10C, CIFAR-100C, and ImageNet-C, which are reconstructed from CIFAR-10 [21], CIFAR-100 [21], and ImageNet [6] with prior shift corruption counterparts [18]. Each dataset contains an image set of 15 corruption style including gaussian noise, shot noise, impulse noise, defocus blur, glass blur, motion blur, zoom blur, snow, frost, fog, brightness, contrast, elastic, pixelated, and jpeg. We follow common protocol in [8,39], evaluate on and report averaged classification error.

Implementation details. For each dataset, we employ a different network, paired as follows: CIFAR-10C to WideResNet-28 [42], CIFAR-100C to ResNeXt-29 [41], and ImageNet-C to ResNet-50 [17]. The networks are optimized in an online manner with batch sizes 200, 200, and 64 and base learning rates $1e-3$, $5e-5$, $2e-4$, respectively. In our approach, we use the same Adam optimizer with $\beta = (0.9,\ 0.999)$ in all experiments. Hyper-parameter $(\lambda,\ \tau)$ is set to $(0.1,\ 1.0)$ in CIFAR10-to-10C, $(1,\ 0.6)$ in CIFAR100-to-100C and $(10,\ 1.0)$ in ImageNet-to-ImageNet-C.

Table 1. Classification error rate (%) for the CIFAR-10C, CIFAR-100C and ImageNet-C online continual test-time adaptation task on the highest corruption severity level 5. For CIFAR-10C the results are evaluated on WideResNet-28, for CIFAR-100C on ResNeXt-29, and for Imagenet-C, ResNet-50 is used. For a fair comparison, we recorded the performance of a single update or no update for all approaches.

	Time			t ⟶															
	Method	Source free	Updates	*Gaussian*	*Shot*	*Impulse*	*Defocus*	*Glass*	*Motion*	*Zoom*	*Snow*	*Frost*	*Fog*	*Brightness*	*Contrast*	*Elastic*	*Pixelate*	*JPEG*	Mean	FLOPs
CIFAR-10C	Source [42]	✓	-	72.3	65.7	72.9	46.9	54.3	34.8	42.0	25.1	41.3	26.0	9.3	46.7	26.6	58.5	30.3	43.5	10.5
	BN-1 [33]	✓	-	28.1	26.1	36.3	12.8	35.3	14.2	12.1	17.3	17.4	15.3	8.4	12.6	23.8	19.7	27.3	20.4	10.5
	TENT-cont. [38]	✓	1	24.8	20.6	28.6	14.4	31.1	16.5	14.1	19.1	18.6	18.6	12.2	20.3	25.7	20.8	24.9	20.7	10.5
	CoTTA [39]	✓	1	**24.3**	21.3	26.6	11.6	27.6	**12.2**	**10.3**	14.8	14.1	**12.4**	**7.5**	10.6	18.3	13.4	17.3	16.2	357.0
	AdaContrast [3]	✓	1	29.1	22.5	30.0	14.0	32.7	14.1	12.0	16.6	14.9	14.4	8.1	**10.0**	21.9	17.7	20.0	18.5	26.3
	Ours	✓	1	24.5	**18.9**	**25.1**	**11.6**	**26.9**	13.2	10.4	**14.2**	**13.4**	12.8	8.4	10.2	**17.6**	**12.4**	**16.5**	**15.7±0.06**	42.0
	GTTA-MIX [28]	✗	1	26.0	21.5	29.7	11.1	30.0	12.2	10.5	15.1	14.1	12.3	7.5	10.0	20.4	15.8	21.4	17.2	42.0
	RMT [8]	✗	1	**21.7**	**18.6**	**24.2**	**10.3**	**24.0**	**11.2**	**9.5**	**12.1**	**11.7**	**10.3**	**7.0**	**8.7**	**14.8**	**10.5**	**14.5**	**13.9**	4252.7
CIFAR-100C	Source [41]	✓	-	73.0	68.0	39.4	29.3	54.1	30.8	28.8	39.5	45.8	50.3	29.5	55.1	37.2	74.7	41.2	46.4	2.16
	BN-1 [33]	✓	-	42.1	40.7	42.7	27.6	41.9	29.7	27.9	34.9	35.0	41.5	26.5	30.3	35.7	32.9	41.2	35.4	2.16
	TENT-cont. [38]	✓	1	**37.2**	35.8	41.7	37.9	51.2	48.3	48.5	58.4	63.7	71.1	70.4	82.3	88.0	88.5	90.4	60.9	2.16
	CoTTA [39]	✓	1	40.1	37.7	39.7	26.9	38.0	27.9	26.4	32.8	31.8	40.3	24.7	26.9	32.5	28.3	**33.5**	32.5	73.34
	AdaContrast [3]	✓	1	42.3	36.8	38.6	27.7	40.1	29.1	27.5	32.9	30.7	38.2	25.9	28.3	33.9	33.3	36.2	33.4	5.4
	Ours	✓	1	39.1	**34.2**	**36.1**	**25.3**	**36.2**	**27.0**	**25.1**	**30.7**	**29.2**	**35.9**	**24.4**	**26.9**	**31.3**	**27.5**	34.8	**30.9±0.09**	8.62
	GTTA-MIX [28]	✗	1	39.4	34.4	36.6	24.7	36.8	26.6	24.3	30.1	28.9	34.6	22.8	25.1	30.7	26.9	34.7	30.4	8.62
	RMT [8]	✗	1	**37.4**	**33.8**	**34.3**	**24.8**	**32.0**	**25.3**	**23.6**	**26.2**	**26.2**	**28.9**	**21.9**	**23.5**	**25.4**	**23.2**	**27.4**	**27.6**	873.72
ImageNet-C	Source [17]	✓	-	97.8	97.1	98.2	81.7	89.8	85.2	78.0	83.5	77.1	75.9	41.3	94.5	82.5	79.3	68.6	82.0	172.26
	BN-1 [33]	✓	-	85.0	83.7	85.0	84.7	84.3	73.7	61.2	66.0	68.2	52.1	**34.9**	82.7	55.9	51.3	59.8	68.6	172.26
	TENT-cont. [38]	✓	1	81.6	74.6	72.7	77.6	73.8	65.5	55.3	61.6	63.0	51.7	38.2	72.1	50.8	47.4	53.3	62.6	8.24
	CoTTA [39]	✓	1	84.7	82.1	80.6	81.3	79.0	68.6	57.5	60.3	60.5	48.3	36.6	**66.1**	**47.2**	**41.2**	**46.0**	62.7	280.3
	AdaContrast [3]	✓	1	82.9	80.9	78.4	81.4	78.7	72.9	64.0	63.5	64.5	53.5	38.4	66.7	54.6	49.4	53.0	65.5	20.6
	Ours	✓	1	**80.4**	**73.8**	**71.2**	**77.5**	**72.9**	**63.9**	**54.1**	**57.9**	**59.8**	**46.3**	35.5	67.2	48.5	44.9	47.2	**60.1±0.14**	32.98
	GTTA-MIX [28]	✗	1	80.5	74.7	72.4	77.8	75.7	64.3	54.0	57.0	58.6	**44.6**	**33.9**	67.5	49.4	44.7	49.3	60.3	32.98
	RMT [8]	✗	1	**77.3**	**73.2**	**71.1**	**73.1**	**71.2**	**61.2**	**53.7**	**54.3**	**58.0**	46.1	38.2	**58.5**	**45.4**	**42.3**	**44.5**	**57.9**	1096.44

Table 2. Classification error rate (%) for the CIFAR-10C, CIFAR-100C, and ImageNet-C online gradual test-time adaptation tasks. We present the performance results in two ways: by calculating the average performance across all severity levels (from level 1 to 5) and by calculating the average performance solely for the highest severity level (level 5). The number in brackets indicates the difference compared to the continual benchmark. For a fair comparison, we recorded the performance of a single update or no update for all approaches.

		Source	BN-1	TENT-cont.	AdaCont.	CoTTA	**Ours**	GTTA-MIX	RMT
	source-free	✓	✓	✓	✓	✓	✓	✗	✗
	Updates	-	-	1	1	1	1	1	1
CIFAR-10C	@level 1-5	24.7	13.7	20.4	12.1	10.9	**9.6**	10.5	**8.1**
	@level 5	43.5	20.4	25.1 (+4.4)	15.8 (-2.7)	14.2 (-2.0)	**11.4** (-4.3)	15.0 (-2.2)	**9.4** (-4.5)
CIFAR-100C	@level 1-5	33.6	29.9	74.8	33.0	26.3	**26.1**	24.3	**23.6**
	@level 5	46.4	35.4	75.9 (+15.0)	35.9 (+2.5)	28.3 (-4.2)	**28.2** (-2.7)	27.6 (-2.8)	**24.3** (-3.2)
ImageNet-C	@level 1-5	58.4	48.3	46.4	66.3	38.8	**38.6**	39.3	**37.8**
	@level 5	82.0	68.6	58.9 (-3.7)	72.6 (+7.1)	43.1 (-19.6)	**41.6** (-18.5)	51.8 (-8.5)	**40.2** (-17.7)

4.2. Continual test-time adaptation

Experimental settings. Similar to TTA setting used in TENT [38], continual test-time adaptation (CTTA) departs from the source pre-trained model. However, while the TTA setting resets the model with source domain parameters after an adaptation to a single target domain, the CTTA considers multiple target domains and has the assumption that does not know when the domain changes. Therefore, the model is adapted to a sequence of test domains in an online manner. Following the CTTA benchmark in [5, 8], the sequence of corruptions consists of the highest severity level 5 of all 15 corruption domains.

Experimental results. Table 1 shows the results for corruption datasets in the continual setting with online measurement. The results show that our FIM-based method achieves comparable and better performance than the prior BN-based approach such as BN [33], and TENT-continual [38] with significant improvement. Our proposed method shows more competitive performance and outperforms most prior works [3, 39]. The comparison to the most recent works [8, 28], our method shows comparable performance despite our method being completely inaccessible to the source domain. Specifically, RMT approach managed to alleviate this augmentation-induced bottleneck, but the computational overhead still

exists due to the intricacies of the training framework itself, involving the use of source prototypes or the source replay process. Moreover, GTTA-MIX introduced intermediate domain mixup methods and showed promising results on CTTA benchmarks, but they still need source datasets for acquiring additional information. To sum up, our proposed approach, requiring no additional frameworks and source information, enables optimization within a singular encoder. Notably, our proposed method achieves results of 15.7%, 30.9%, and 60.1% for CIFAR-10C, CIFAR-100C, and ImageNet-C, respectively. These results are comparable to those obtained with computationally intensive methods such as CoTTA [39] and demonstrate performance on par with GTTA-MIX and RMT in non-source-free settings. For additional experimental results on model variation, please refer to Supplementary Sec. A.

Table 3. Comparison between AutoRGN and our method for mean classification error (%). The number in brackets denotes the performance difference compared to TENT continual [38].

Method	CIFAR-10C	CIFAR-100C	ImageNet-C
TENT cont. [38]	20.7	60.9	62.6
AutoRGN [23]	18.06 (-2.64)	33.16 (-27.74)	62.24 (-0.36)
Ours	**15.74** (-4.69)	**30.91** (-29.99)	**60.07** (-2.53)

Comparison with AutoRGN. We further compare our approach with AutoRGN [23] using the same backbone on CTTA benchmark datasets to ensure a fair comparison. The quantitative results are presented in Table 3. AutoRGN outperforms TENT in the layer-wise learning approach of CTTA, achieving accuracy rates of 18.06%, 33.16%, and 62.24% for CIFAR-10C, CIFAR-100C, and ImageNet-C. Although AutoRGN improves CTTA's performance compared to the baseline, our approach maintains a lower mean error.

4.3. Gradual test-time adaptation

Experimental settings. We also evaluate our proposed method on the gradual test-time adaptation (GTTA) benchmarks with mean classification error for CIFAR-10C, CIFAR-100C, and ImageNet-C datasets. While the CTTA online setting encounters the highest corruption domains sequentially, GTTA online setting encounters gradually increasing severity sequentially as follows:

$$\underbrace{\cdots \to 2 \to 1}_{\text{Domain } t-1} \xrightarrow[\text{shift}]{\text{domain}} \underbrace{1 \to 2 \to \cdots \to 5 \to \cdots \to 1}_{\text{Domain } t\text{, gradually changing severity}} \xrightarrow[\text{shift}]{\text{domain}}.$$

Experimental results. We present the mean classification error results for GTTA in Table 2, encompassing all severity levels from 1 to 5, as well as reporting specifically on level 5. Previous approaches, including TENT-continual [38] and AdaContrast [3], exhibit degradation on specific datasets. Although EMA baselines like CoTTA [39] and RMT [8] demonstrate performance enhancements for GTTA [28], the computational complexity and source dependency become bottlenecks during test-time adaptation. Our method, which employs layer-wise learning rates to update each parameter, yields competitive results with these approaches and notably outperforms the TENT continual baseline. Furthermore, our method shows improvement in mean error for severity level 5, with 11.4% on CIFAR10-to-10C, 28.2% on CIFAR100-to-100C and 41.6% on ImageNet-C datasets, compared to CoTTA and GTTA-MIX.

4.4. Ablation study and Analysis

To further analyze and validate the components of our method, we conduct ablation experiments on three benchmark datasets in the continual test-time adaptation (CTTA) setting. Additionally, to demonstrate the impact of the Fisher information matrix across various domain shifts, we provide a graph depicting layer-wise learning weight.

Contributions of objective. We conduct additional experiments to validate the effect of the loss function in Eq. (9) and its corresponding hyper-parameter λ, i.e., $\lambda \in \{0, 0.01, 0.1, 1.0, 10, 100\}$. In Table 4, we report the mean classification error rates on three corruption datasets with CTTA setting. As we can see, the best performance is achieved at different values of λ, indicating that entropy minimization loss should be balanced with respect to the number of classes. In the case where $\lambda = 0$, entropy minimization is solely used as an optimization function and still shows improvement. For CIFAR-10C, solely optimized with entropy minimization and FIM learning weight improves performance about 3.5% and with hyperparameter tuning, the best setting ($\lambda = 0.1$) improves performance about 4.9% from the baseline [38]. In the case of CIFAR-100C, solely optimized with entropy minimization and FIM learning weight shows remarkable improvement in performance about 28.6% and with hyperparameter tuning, the best setting ($\lambda = 1.0$) improves performance about 30.0% from the baseline [38]. And for ImageNet-C, solely optimized with entropy minimization and FIM learning weight improves performance about 0.2%, and with hyperparameter tuning, the best setting ($\lambda = 10$) improves performance about 2.4% from the baseline [38]. Here it indicates that our proposed FIM-based learning weight method shows improvements in all datasets even without the consistency loss in Eq. (8) and with hyperparameter tuning for consistency loss function, we can further improve performance.

Table 4. Mean classification error (%) with varing λ.

λ	0	0.01	0.1	1.0	10	100
CIFAR-10C	17.22	16.65	**15.74**	17.76	19.04	17.93
CIFAR-100C	32.29	32.40	32.23	**30.91**	31.66	31.74
ImageNet-C	62.38	62.33	62.24	61.08	**60.07**	60.95

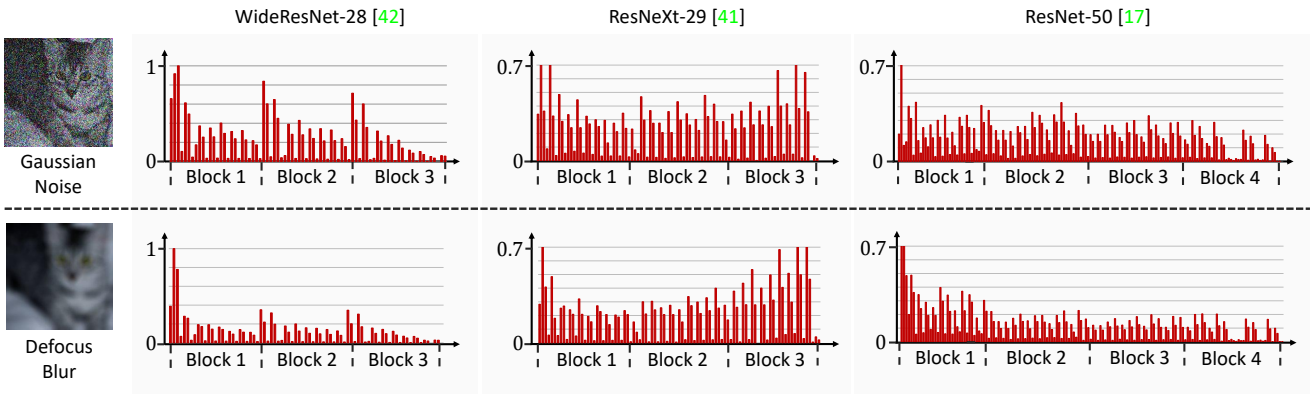


Figure 3. Results of ablation are presented, illustrating the normalized layer learning weight discrepancy between the dataset domain and each model. The first row displays the effects of Gaussian noise corruption at severity level 5, while the second row depicts Defocus blur corruption at severity level 5. The first column represents the CIFAR10-C dataset, the second column the CIFAR100-C dataset, and the last column the ImageNet-C dataset.

Exponential min-max scaler. Also, we report the image classification error rate in Table 5 according to the scaler hyperparameter τ, i.e., $\tau \in \{0.6, 0.7, 0.8, 0.9, 1.0, 1.1, 1.2\}$. From the results, we demonstrate that we can achieve remarkable progress in performance in each dataset by simply applying τ to learning weight in Eq. (7). For CIFAR-10C and ImageNet-C, $\tau = 1.0$ shows the best performance, which indicates that the original learning weight with min-max normalization is sufficient. Meanwhile for CIFAR-100C, $\tau = 0.6$ shows the best performance. We present additional τ variation results in Supplementary Sec. B.2.

Table 5. Mean classification error (%) with varing τ.

τ	0.6	0.7	0.8	0.9	1.0	1.1	1.2
CIFAR-10C	39.55	19.96	17.74	16.72	**15.74**	15.97	16.20
CIFAR-100C	**30.91**	31.42	32.16	32.42	32.76	33.03	33.30
ImageNet-C	77.29	69.89	63.44	61.08	**60.07**	61.21	61.65

Layer-wise learning weight in domain shifts. In Figure 3, we show two corruption domains (gaussian noise, defocus blur) and their corresponding layer-wise learning weights using Eq. (7) with $\tau = 1$. The learning weights differ across layers based on corruption types and networks. Considering frequency components in corruption domains, variations appear in deeper layers of a hierarchical deep neural network (DNN). Domains with reduced high-frequency components, like defocus blur, tend to concentrate weights in initial layers. In contrast, noise-type domains, retaining higher frequency components, show more evenly distributed weights across network layers. These tendencies are apparent in domain-specific weight distributions. These results demonstrate our method's robustness for test-time adaptation across diverse domains. In Supplementary Sec. B.3 and Sec. B.4, we provide visualizations of the diagonal FIM and additional discussion regarding layer-wise learning weights.

5. Conclusion and Future Work

This paper proposes a simple yet effective layer-wise auto-weighting algorithm that enhances non-stationary Test-Time Adaptation (TTA) performance. With considerably less computational load, our method is more suitable for edge devices requiring online adaptation. First, we leverage the Fisher Information Matrix (FIM) to autonomously identify the layers that need preservation or concentrated adaptation. Second, we introduce an exponential min-max scaler to amplify the differences in learning weights across layers. This approach results in certain layers becoming nearly frozen while mitigating the distortion of learning weight outliers. As a result, our method selectively focuses on layers associated with log-likelihood changes in the target domain while preserving the unrelated ones, minimizing catastrophic forgetting and error accumulation. Our FIM-based weighted learning leads to more efficient adaptation to non-stationary target domains. Through extensive experiments and ablation studies on various benchmarks and networks, we have demonstrated that our method outperforms conventional CTTA approaches while significantly reducing the computational load. In this context, we believe that our work contributes to making test-time adaptation for edge devices more feasible in practice and hope that it will inspire further research in this area.

We also intend to employ our method in diverse recognition tasks, including segmentation and object detection, where a higher level of output granularity is needed.

Acknowledgement. This research was supported by the Yonsei Signature Research Cluster Program of 2022 (2022-22-0002) and the KIST Institutional Program (Project No.2E31051-21-203).

References

[1] Malik Boudiaf, Romain Mueller, Ismail Ben Ayed, and Luca Bertinetto. Parameter-free online test-time adaptation. In *CVPR*, 2022. 2

[2] Niladri S Chatterji, Behnam Neyshabur, and Hanie Sedghi. The intriguing role of module criticality in the generalization of deep networks. In *ICLR*, 2020. 2, 3

[3] Dian Chen, Dequan Wang, Trevor Darrell, and Sayna Ebrahimi. Contrastive test-time adaptation. In *CVPR*, 2022. 2, 4, 6, 7

[4] Sungha Choi, Seunghan Yang, Seokeon Choi, and Sungrack Yun. Improving test-time adaptation via shift-agnostic weight regularization and nearest source prototypes. In *ECCV*, 2022. 2

[5] Francesco Croce, Maksym Andriushchenko, Vikash Sehwag, Edoardo Debenedetti, Nicolas Flammarion, Mung Chiang, Prateek Mittal, and Matthias Hein. Robustbench: a standardized adversarial robustness benchmark. In *NeurIPS*, 2021. 6

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 5

[7] Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. Sharp minima can generalize for deep nets. In *ICML*, 2017. 2, 4

[8] Mario Döbler, Robert A Marsden, and Bin Yang. Robust mean teacher for continual and gradual test-time adaptation. In *CVPR*, 2023. 1, 2, 3, 5, 6, 7

[9] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2020. 1

[10] Utku Evci, Vincent Dumoulin, Hugo Larochelle, and Michael C Mozer. Head2toe: Utilizing intermediate representations for better transfer learning. In *ICML*, 2022. 3

[11] Yossi Gandelsman, Yu Sun, Xinlei Chen, and Alexei Efros. Test-time training with masked autoencoders. *NeurIPS*, 2022. 2

[12] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *CVPR*, 2016. 2

[13] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *ICML*, 2017. 2, 3

[14] Yunhui Guo, Honghui Shi, Abhishek Kumar, Kristen Grauman, Tajana Rosing, and Rogerio Feris. Spottune: transfer learning through adaptive fine-tuning. In *CVPR*, 2019. 3

[15] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *CVPR*, 2022. 1

[16] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020. 1

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1, 4, 5, 6

[18] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *ICLR*, 2019. 5

[19] Jin Kim, Jiyoung Lee, Jungin Park, Dongbo Min, and Kwanghoon Sohn. Pin the memory: Learning to generalize semantic segmentation. In *CVPR*, 2022. 1

[20] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 2017. 3

[21] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 5

[22] Jaejun Lee, Raphael Tang, and Jimmy Lin. What would elsa do? freezing layers during transformer fine-tuning. *arXiv preprint arXiv:1911.03090*, 2019. 3

[23] Yoonho Lee, Annie S Chen, Fahim Tajwar, Ananya Kumar, Huaxiu Yao, Percy Liang, and Chelsea Finn. Surgical fine-tuning improves adaptation to distribution shifts. *ICLR*, 2023. 2, 3, 7

[24] Jian Liang, Dapeng Hu, and Jiashi Feng. Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation. In *ICML*, 2020. 2

[25] Yuhan Liu, Saurabh Agarwal, and Shivaram Venkataraman. Autofreeze: Automatically freezing model blocks to accelerate fine-tuning. *arXiv preprint arXiv:2102.01386*, 2021. 3

[26] Yuejiang Liu, Parth Kothari, Bastien Van Delft, Baptiste Bellot-Gurlet, Taylor Mordan, and Alexandre Alahi. Ttt++: When does self-supervised test-time training fail or thrive? In *NeurIPS*, 2021. 2

[27] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. Unsupervised domain adaptation with residual transfer networks. *NeurIPS*, 2016. 1

[28] Robert A Marsden, Mario Döbler, and Bin Yang. Gradual test-time adaptation by self-training and style transfer. In *ICLR*, 2022. 3, 6, 7

[29] Behnam Neyshabur, Hanie Sedghi, and Chiyuan Zhang. What is being transferred in transfer learning? In *NeurIPS*, 2020. 2, 3

[30] Vinay V Ramasesh, Ethan Dyer, and Maithra Raghu. Anatomy of catastrophic forgetting: Hidden representations and task semantics. In *ICLR*, 2020. 3

[31] Amélie Royer and Christoph Lampert. A flexible selection scheme for minimum-effort transfer learning. In *WACV*, 2020. 3

[32] Levent Sagun, Utku Evci, V Ugur Guney, Yann Dauphin, and Leon Bottou. Empirical analysis of the hessian of over-parametrized neural networks. 2017. 2, 4

[33] Steffen Schneider, Evgenia Rusak, Luisa Eck, Oliver Bringmann, Wieland Brendel, and Matthias Bethge. Improving robustness against common corruptions by covariate shift adaptation. In *NeurIPS*, 2020. 2, 6

[34] Jonathan Schwarz, Wojciech Czarnecki, Jelena Luketina, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress & compress: A scalable framework for continual learning. In *ICML*, 2018. 3, 4

[35] Claude Elwood Shannon. A mathematical theory of communication. *The Bell system technical journal*, 1948. 2

[36] Kihyuk Sohn, David Berthelot, Nicholas Carlini, Zizhao Zhang, Han Zhang, Colin A Raffel, Ekin Dogus Cubuk, Alexey Kurakin, and Chun-Liang Li. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. In *NeurIPS*, 2020. 4

[37] Hong Hui Tan and K. Lim. Review of second-order optimization techniques in artificial neural networks backpropagation. *ACME*, 2019. 3, 4

[38] Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. In *ICLR*, 2021. 1, 2, 5, 6, 7

[39] Qin Wang, Olga Fink, Luc Van Gool, and Dengxin Dai. Continual test-time domain adaptation. In *CVPR*, 2022. 2, 5, 6, 7

[40] Guoqiang Wei, Cuiling Lan, Wenjun Zeng, and Zhibo Chen. Metaalign: Coordinating domain alignment and classification for unsupervised domain adaptation. In *CVPR*, 2021. 1, 2

[41] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *CVPR*, 2017. 4, 5, 6

[42] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *BMVC*, 2016. 4, 5, 6

[43] Chiyuan Zhang, Samy Bengio, and Yoram Singer. Are all layers created equal? *JMLR*, 2022. 2, 3

[44] Marvin Zhang, Sergey Levine, and Chelsea Finn. Memo: Test time robustness via adaptation and augmentation. In *NeurIPS*, 2022. 2