# A Layer Selection Approach to Test Time Adaptation

**Sabyasachi Sahoo**[12], **Mostafa ElAraby**[23], **Jonas Ngnawe**[12], **Yann Batiste Pequignot**[1], **Frederic Precioso**[4], **Christian Gagné**[125]
[1]IID, Université Laval [2]Mila [3]Université de Montréal
[4]Université Cote d'Azur, CNRS, INRIA, I3S, Maasai [5]Canada CIFAR AI Chair

## Abstract

Test Time Adaptation (TTA) addresses the problem of distribution shift by adapting a pretrained model to a new domain during inference. When faced with challenging shifts, most methods collapse and perform worse than the original pretrained model. In this paper, we find that not all layers are equally receptive to the adaptation, and the layers with the most misaligned gradients often cause performance degradation. To address this, we propose GALA, a novel layer selection criterion to identify the most beneficial updates to perform during test time adaptation. This criterion can also filter out unreliable samples with noisy gradients. Its simplicity allows seamless integration with existing TTA loss functions, thereby preventing degradation and focusing adaptation on the most trainable layers. This approach also helps to regularize adaptation to preserve the pretrained features, which are crucial for handling unseen domains. Through extensive experiments, we demonstrate that the proposed layer selection framework improves the performance of existing TTA approaches across multiple datasets, domain shifts, model architectures, and TTA losses.

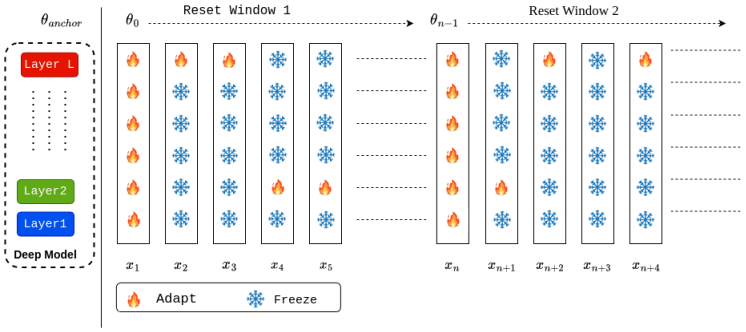



Figure 1: Gradient-Aligned Layer Adaptation or GALA framework adapts the most gradient-aligned layer per sample. It adapts all the layers for the first sample in a reset window (e.g., $x_1, x_n, \dots$). For all the other samples, it adapts the most gradient-aligned layer per sample. It can also skip the adaptation on a given sample if all the layers are misaligned. We use a reset window to periodically reset the anchor parameters to allow for a change in direction.

## 1 Introduction

Test Time Adaptation (TTA) [1] has emerged as a promising approach for addressing the problem of data distribution shifts [2] by adapting pretrained models to unseen domains at inference. However, these methods often falter when confronted with severe or diverse distributional changes. Moreover, the selection of layers in existing approaches typically remains unchanged across different shifts [3], and optimal layer selection remains largely unexplored in the context of TTA.

38th Workshop on Fine-Tuning in Machine Learning (NeurIPS 2024).

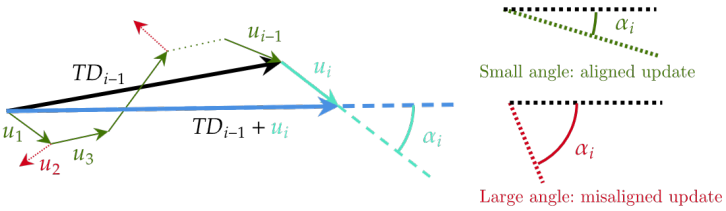



Figure 2: Illustration of proposed criterion based on angular deviation. Different layers can be ranked based on their alignments with previous gradient updates. In the figure, updates drawn in red are discarded, while green updates are applied, adding up to $\mathbf{TD}_{i-1}$. The update under scrutiny $\mathbf{u_i}$ is drawn in cyan, and its sum with $\mathbf{TD}_{i-1}$ is drawn in blue. Application of update $\mathbf{u_i}$ or not is based on the angle $\alpha_i$.

The contributions of our work are summarized as follows: **(1)** We study the problem of layer selection for TTA and find that while adapting specific layers can enhance performance, the optimal set of layers for adaptation is not universal but rather contingent upon the particular distribution shift encountered and the TTA loss function employed during inference. **(2)** We introduce Gradient-Aligned Layer Adaptation (GALA), a novel layer selection criterion to identify good layers to adapt per sample that can be applied across various distribution shifts and TTA loss functions at test time. **(3)** Through extensive experiments across different backbones, datasets, and TTA losses, we show that GALA outperforms standard (*ERM*/no-adaptation, *All layers*) and existing layer selection baselines (AutoRGN, AutoSNR) used in finetuning [4].

## 2 Proposed Approach

### 2.1 Layer Selection Framework for TTA

Let $f_{\theta_{src}}$ be the model parameterized by $\theta_{src}$, trained on the source domain $\mathcal{D}_{src}$. At test time, TTA adapts the model to obtain $\theta_i$ for each incoming target sample $x_i \sim \mathcal{D}_{tgt}$ using an update equation

$$\theta_{i,l} = \theta_{i-1,l} + \mathbf{u}_{i,l}. \tag{1}$$

For SGD with TTA loss $\mathcal{L}$, $\mathbf{u}_i = -\eta \nabla \mathcal{L}(x_i; \theta_{i-1})$. We propose a layer-wise adaptation by introducing a binary mask $m_{i,l} \in \{0, 1\}$ to the update $\mathbf{u}_{i,l}$ of each layer $l$ (*c.f.* Fig. 1):

$$\theta_{i,l} = \theta_{i-1,l} + m_{i,l}\, \mathbf{u}_{i,l}. \tag{2}$$

### 2.2 Cosine distance criterion

Let us consider the total displacement

$$\mathbf{TD}_{i-1,l} = \sum_{j=1}^{i-1} m_{j,l}\mathbf{u}_{j,l} = \theta_{i-1,l} - \theta_{0,l}, \tag{3}$$

which is in the same direction as the average of all gradients so far (Fig. 2). We define the mask $m_{i,l}$ based on the cosine distance $\cos(\alpha_{i,l})$ between the current update $\mathbf{u}_{i,l}$ and the anticipated total displacement $\mathbf{TD}_{i-1,l} + \mathbf{u}_{i,l}$ as

$$\cos(\alpha_{i,l}) = \frac{\mathbf{u}_{i,l} \cdot (\mathbf{u}_{i,l} + \mathbf{TD}_{i-1,l})}{\|\mathbf{u}_{i,l}\|_2 \, \|\mathbf{u}_{i,l} + \mathbf{TD}_{i-1,l}\|_2}. \tag{4}$$

Layers with updates aligned with the total displacement are selected for adaptation (*c.f.* Fig. 2)

$$m_{i,l} = \begin{cases} 1 & \text{if } \cos(\alpha_{i,l}) > \lambda \\ 0 & \text{otherwise} \end{cases}, \tag{5}$$

where $\lambda$ is the selection threshold.

### 2.3 Cosine distance with reset

To address cases where the gradient update trajectory needs to change direction, we propose a reset mechanism that updates the anchor point $\theta_{r,l}$ every $s$ step,

$$\mathbf{TD}_{i,l} = \theta_{i,l} - \theta_{r,l}, \tag{6}$$

where $r = \lfloor \frac{i-1}{s} \rfloor$ (*c.f.* Fig. 1).

Table 1: Accuracy (%) of various layer selection methods on Domainbed benchmark (setup described in Sec. 3). The best method for a given TTA loss and backbone is in bold.

| | TTA | Method | PACS ↑ | VLCS ↑ | Terra ↑ | Office ↑ | Mean ↑ |
|---|---|---|---|---|---|---|---|
| ResNet-18 | ERM | – | 80.99 (±0.9) | 75.14 (±1.2) | 40.80 (±0.2) | 62.18 (±0.4) | 64.78 |
| | PL | All layers | 81.79 (±0.7) | 65.69 (±1.5) | 35.40 (±9.7) | 60.20 (±1.4) | 60.77 |
| | | AutoRGN | 82.82 (±0.6) | 72.63 (±1.3) | 38.18 (±6.1) | 62.38 (±0.2) | 64.00 |
| | | AutoSNR | 80.58 (±1.2) | 65.72 (±1.8) | 35.01 (±10.4) | 59.82 (±0.9) | 60.28 |
| | | GALA | **83.56 (±0.6)** | **75.48 (±1.2)** | **44.19 (±1.1)** | **62.67 (±0.2)** | **66.47** |
| | SHOT | All layers | 83.48 (±0.3) | 66.23 (±2.8) | 33.81 (±1.3) | 63.03 (±0.4) | 61.64 |
| | | AutoRGN | **84.10 (±0.5)** | 69.78 (±1.3) | 37.37 (±0.7) | 63.09 (±0.2) | 63.59 |
| | | AutoSNR | 83.43 (±0.3) | 66.26 (±2.7) | 33.75 (±1.2) | 63.02 (±0.4) | 61.62 |
| | | GALA | 83.92 (±0.8) | **76.23 (±1.1)** | **42.13 (±1.4)** | **63.32 (±0.3)** | **66.40** |
| ResNet-50 | ERM | – | 82.84 (±0.5) | 75.83 (±0.9) | 46.14 (±2.3) | 66.93 (±0.3) | 67.93 |
| | PL | All layers | 82.36 (±2.8) | 69.22 (±1.4) | 42.28 (±3.2) | 61.54 (±3.3) | 63.85 |
| | | AutoRGN | **85.03 (±1.9)** | 75.35 (±1.4) | 48.44 (±2.4) | 66.93 (±0.3) | 68.94 |
| | | AutoSNR | 83.41 (±3.4) | 70.14 (±4.6) | 44.08 (±3.4) | 61.95 (±3.0) | 64.90 |
| | | GALA | 84.87 (±0.8) | **76.88 (±1.6)** | **50.10 (±2.5)** | **67.34 (±0.3)** | **69.80** |
| | SHOT | All layers | 85.15 (±1.1) | 64.25 (±1.1) | 35.33 (±3.1) | 67.37 (±0.3) | 63.03 |
| | | AutoRGN | **86.34 (±1.1)** | 70.2 (±0.9) | 40.59 (±1.3) | 68.10 (±0.4) | 66.31 |
| | | AutoSNR | 85.51 (±0.5) | 64.26 (±1.3) | 34.97 (±3.2) | 67.33 (±0.2) | 63.02 |
| | | GALA | 86.13 (±0.8) | **76.48 (±1.0)** | **45.94 (±1.6)** | **68.13 (±0.3)** | **69.17** |

## 3 Experiments

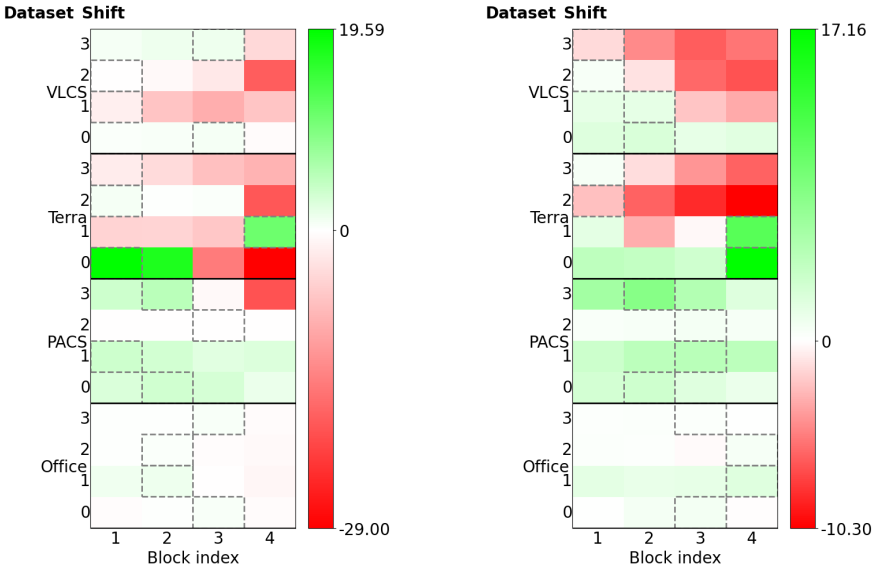
**Experimental Setup** We utilize the Domainbed [2] benchmark and adhere to the evaluation protocol outlined in [5]. Our evaluation spans two small shift (VLCS [6], Terra Incognita [7]) and two large shift datasets (PACS [8], Office-Home [9]). Pretrained models are obtained by training on source domains using ResNet-18 or ResNet-50 backbones, following the pretraining protocol in [2]. At test time, the pretrained models are adapted to samples from the target domain using two TTA losses, Pseudo-Labeling (PL) [10] and SHOT [11], with its hyperparameters tuned according to [12]. We compare GALA against two standard (ERM/no-adaptation, *All layers*) and two popular layer selection baselines (AutoRGN, AutoSNR [4]). Results for GALA are reported with a *window size* ($s$) of 20 and a *selection threshold* ($\lambda$) of 0.75 with *single-layer* granularity of adaptation.

**Results** Key takeaways from results reported in Tab. 1: GALA outperforms ERM by 2% overall and *All layers* TTA baselines by over 5% across all losses, backbones, and datasets. Existing baselines like AutoRGN and AutoSNR improve performance compared to *All layers* TTA in most setups but fail against ERM for some datasets and TTA losses. GALA consistently shows superior performance across all datasets and TTA losses, achieving an overall improvement of about 2%. On large shift datasets, GALA outperforms ERM to enhance performance, similar to layer selection baselines. But on small shift datasets, while existing baselines struggle against the ERM baseline, GALA prevents degradation from over-adaptation and consistently outperforms ERM.

## 4 Layer Selection Study

**Layer Selection matters.** We conducted a layer selection study to examine the impact of adapting a single block of layers while freezing others. Figure 3 shows the difference in accuracy between adaptation (TTA) and no adaptation (ERM) across all blocks for each TTA loss and dataset shift in Domainbed. We found that not all layers are equally receptive to adaptation; some layers benefit one shift but may be detrimental to another, depending on the loss used. Identifying the right layer to adapt at test time is challenging. To address this, we propose GALA, a layer selection criterion for TTA that effectively identifies suitable layers for adaptation, consistently improving over existing baselines on the Domainbed benchmark.

**How do good layers differ from bad layers?** The Tiny-Domainbed benchmark, a smaller version of Domainbed, was created to analyze per-layer differences, focusing on critical shifts with the brightest red/green layers. Tab. 2 shows that adaptation with the *Worst Block* results in poor TTA accuracy and higher forgetting, while the *Best Block* leads to better generalization and reduced




(a) PL loss-based TTA (b) SHOT loss-based TTA

Figure 3: Heatmap of Performance improvement (%) per-block on Domainbed benchmark. Performance improvement is the difference between the TTA accuracy of a given block/layer and ERM accuracy for the same shift. Positive performance improvements are shown in green, and negative performance improvements (or degradation) are in red. Using the bounding box, we highlight the best block per loss and dataset shift. Further details in Sec. 4.

Table 2: Effect of various layer selection methods on TTA Accuracy (%), Generalization (%), Forgetting (%) and Spearman correlation with Best Block ($\in [-1, 1]$) averaged over different shifts on Tiny-Domainbed benchmark (with the setup described in Sec. 4). *TTA Acc* is the accuracy of testing samples from the target domain seen during adaptation. *Generalization* is the accuracy of the held-out split of the target domain after adaptation. *Forgetting* is the drop in accuracy on the held-out split of source domains after adaptation. *Rank correlation* is the Spearman correlation of layer selection rank between the oracle and the method. Bold and underlined denote best and second-best, respectively.

| **Method** | **TTA Acc.** ↑ | **Gen.** ↑ | **Forget.** ↓ | **Rank corr.** ↑ |
|---|---|---|---|---|
| All Blocks | 53.6 | 46.5 | 31.3 | N/A |
| Worst Block (oracle) | 43.5 | 38.7 | 39.9 | -1 |
| Best Block (oracle) | **64.1** | **63.9** | 28.7 | **1** |
| Random Block | 53.1 | 49.1 | <u>13.1</u> | 0 |
| GALA | <u>59.4</u> | <u>58.0</u> | **9.3** | <u>0.76</u> |

source forgetting. This suggests that good layers can learn target domain features more effectively. The proposed GALA method aims to identify good layers for better adaptation and reduced source forgetting.

**How does GALA compare to oracle strategies?** To analyze GALA's layer selection behavior, we compare it to the oracle strategies of Best block and Worst block on the Tiny-Domainbed benchmark (Tab. 2). GALA substantially improves over All Blocks, Worst Block, and Random Block methods. The Best Block method acts as an empirical upper-bound performance if we have access to a target domain with labels, while GALA comes close to this performance without requiring any target labels. GALA selects layers with the most aligned gradients and can stop adaptation if gradients are noisy, preventing degradation. It tends to select blocks with better accuracy, showing a good correlation with the oracle TTA performance.

## References


[1] Jian Liang, R. He, and Tien-Ping Tan. A comprehensive survey on test-time adaptation under distribution shifts. *International Journal of Computer Vision*, 2023.

[2] Ishaan Gulrajani and David Lopez-Paz. In search of lost domain generalization. In *International Conference on Learning Representations*, 2021.

[3] Zixin Wang, Yadan Luo, Liang Zheng, Zhuoxiao Chen, Sen Wang, and Zi Huang. In search of lost online test-time adaptation: A survey. *International Journal of Computer Vision (IJCV)*, 2024.

[4] Yoonho Lee, Annie S Chen, Fahim Tajwar, Ananya Kumar, Huaxiu Yao, Percy Liang, and Chelsea Finn. Surgical fine-tuning improves adaptation to distribution shifts. In *The Eleventh International Conference on Learning Representations*, 2023.

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

[6] Chen Fang, Ye Xu, and Daniel N Rockmore. Unbiased metric learning: On the utilization of multiple datasets and web images for softening bias. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1657–1664, 2013.

[7] Sara Beery, Grant Van Horn, and Pietro Perona. Recognition in terra incognita. In *Proceedings of the European conference on computer vision (ECCV)*, pages 456–473, 2018.

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

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

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

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

[12] Hao Zhao, Yuejiang Liu, Alexandre Alahi, and Tao Lin. On pitfalls of test-time adaptation. In *International conference on machine learning*. PMLR, 2023.

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

[14] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *International Conference on Learning Representations*, 2019.

[15] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *British Machine Vision Conference*. British Machine Vision Association, 2016.

[16] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017.

[17] Francesco Croce, Maksym Andriushchenko, Vikash Sehwag, Edoardo Debenedetti, Nicolas Flammarion, Mung Chiang, Prateek Mittal, and Matthias Hein. Robustbench: a standardized adversarial robustness benchmark. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

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

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

[20] Minguk Jang, Sae-Young Chung, and Hye Won Chung. Test-time adaptation via self-training with nearest neighbor information. In *The Twelfth International Conference on Learning Representations*, 2023.

[21] Xuefeng Hu, Ke Zhang, Min Sun, Albert Chen, Cheng-Hao Kuo, and Ram Nevatia. Bafta: Backprop-free test-time adaptation for zero-shot vision-language models. *arXiv preprint arXiv:2406.11309*, 2024.

[22] Yongyi Su, Xun Xu, and Kui Jia. Revisiting realistic test-time training: Sequential inference and adaptation by anchored clustering. *Advances in Neural Information Processing Systems*, 35:17543–17555, 2022.

[23] Ziyi Zhang, Weikai Chen, Hui Cheng, Zhen Li, Siyuan Li, Liang Lin, and Guanbin Li. Divide and contrast: Source-free domain adaptation via adaptive contrastive learning. *Advances in Neural Information Processing Systems*, 35:5137–5149, 2022.

[24] Jin Gao, Jialing Zhang, Xihui Liu, Trevor Darrell, Evan Shelhamer, and Dequan Wang. Back to the source: Diffusion-driven adaptation to test-time corruption. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11786–11796, 2023.

[25] Marvin Zhang, Sergey Levine, and Chelsea Finn. Memo: Test time robustness via adaptation and augmentation. *Advances in Neural Information Processing Systems*, 35:38629–38642, 2022.

[26] Bowen Zhao, Chen Chen, and Shu-Tao Xia. Delta: Degradation-free fully test-time adaptation. In *The Eleventh International Conference on Learning Representations*, 2023.

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

[28] Juhyeon Shin, Jonghyun Lee, Saehyung Lee, Minjun Park, Dongjun Lee, Uiwon Hwang, and Sungroh Yoon. Gradient alignment with prototype feature for fully test-time adaptation. *arXiv preprint arXiv:2402.09004*, 2024.

[29] Chanho Ahn, Eunwoo Kim, and Songhwai Oh. Deep elastic networks with model selection for multi-task learning. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6529–6538, 2019.

[30] Pengsheng Guo, Chen-Yu Lee, and Daniel Ulbricht. Learning to branch for multi-task learning. In *International conference on machine learning*, pages 3854–3863. PMLR, 2020.

[31] Ximeng Sun, Rameswar Panda, Rogerio Feris, and Kate Saenko. Adashare: Learning what to share for efficient deep multi-task learning. *Advances in Neural Information Processing Systems*, 33:8728–8740, 2020.

[32] Matthew Wallingford, Hao Li, Alessandro Achille, Avinash Ravichandran, Charless Fowlkes, Rahul Bhotika, and Stefano Soatto. Task adaptive parameter sharing for multi-task learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7561–7570, 2022.

[33] Jathushan Rajasegaran, Munawar Hayat, Salman H Khan, Fahad Shahbaz Khan, and Ling Shao. Random path selection for continual learning. *Advances in neural information processing systems*, 32, 2019.

[34] Amelia Sorrenti, Giovanni Bellitto, Federica Proietto Salanitri, Matteo Pennisi, Concetto Spampinato, and Simone Palazzo. Selective freezing for efficient continual learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3550–3559, 2023.

[35] Haiyan Zhao, Tianyi Zhou, Guodong Long, Jing Jiang, and Chengqi Zhang. Does continual learning equally forget all parameters? In *International Conference on Machine Learning*, pages 42280–42303. PMLR, 2023.

[36] Andrew Lee, Xiaoyan Bai, Itamar Pres, Martin Wattenberg, Jonathan K. Kummerfeld, and Rada Mihalcea. A mechanistic understanding of alignment algorithms: A case study on DPO and toxicity. In *Forty-first International Conference on Machine Learning*, 2024.

[37] Wei Zhang, Chaoqun Wan, Yonggang Zhang, Yiu-ming Cheung, Xinmei Tian, Xu Shen, and Jieping Ye. Interpreting and improving large language models in arithmetic calculation. In *Forty-first International Conference on Machine Learning*, 2024.

[38] Ziqian Lin, Sreya Dutta Roy, and Yixuan Li. Mood: Multi-level out-of-distribution detection. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, pages 15313–15323, 2021.

[39] Yeonguk Yu, Sungho Shin, Seongju Lee, Changhyun Jun, and Kyoobin Lee. Block selection method for using feature norm in out-of-distribution detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15701–15711, 2023.

[40] Maxime Darrin, Guillaume Staerman, Eduardo Dadalto Câmara Gomes, Jackie CK Cheung, Pablo Piantanida, and Pierre Colombo. Unsupervised layer-wise score aggregation for textual ood detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17880–17888, 2024.

[41] Yue Yuan, Rundong He, Yicong Dong, Zhongyi Han, and Yilong Yin. Discriminability-driven channel selection for out-of-distribution detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 26171–26180, 2024.

[42] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. *arXiv preprint arXiv:1803.07728*, 2018.

[43] Florian Bordes, Randall Balestriero, Quentin Garrido, Adrien Bardes, and Pascal Vincent. Guillotine regularization: Why removing layers is needed to improve generalization in self-supervised learning. *Transactions on Machine Learning Research*, 2023.

[44] Ankita Pasad, Bowen Shi, and Karen Livescu. Comparative layer-wise analysis of self-supervised speech models. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2023.

[45] Prithvijit Chattopadhyay, Yogesh Balaji, and Judy Hoffman. Learning to balance specificity and invariance for in and out of domain generalization. In *European Conference in Computer Vision (ECCV)*, 2020.

[46] Yunhui Guo, Honghui Shi, Abhishek Kumar, Kristen Grauman, Tajana Rosing, and Rogerio Feris. Spottune: transfer learning through adaptive fine-tuning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4805–4814, 2019.

[47] Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International Conference on Machine Learning*, pages 23965–23998. PMLR, 2022.

[48] Abhishek Panigrahi, Nikunj Saunshi, Haoyu Zhao, and Sanjeev Arora. Task-specific skill localization in fine-tuned language models. In *International conference on machine learning*. PMLR, 2023.

[49] Polina Kirichenko, Pavel Izmailov, and Andrew Gordon Wilson. Last layer re-training is sufficient for robustness to spurious correlations. In *The Eleventh International Conference on Learning Representations*, 2023.

[50] Nihal Murali, Aahlad Puli, Ke Yu, Rajesh Ranganath, and Kayhan Batmanghelich. Beyond distribution shift: Spurious features through the lens of training dynamics. *Transactions on machine learning research*, 2023, 2023.

[51] Pedro Vianna, Muawiz Sajjad Chaudhary, An Tang, Guy Cloutier, Guy Wolf, Michael Eickenberg, and Eugene Belilovsky. Channel selection for test-time adaptation under distribution shift. *NeurIPS 2023 Workshop on Distribution Shifts: New Frontiers with Foundation Models*, 2023.

[52] Junyoung Park, Jin Kim, Hyeongjun Kwon, Ilhoon Yoon, and Kwanghoon Sohn. Layer-wise auto-weighting for non-stationary test-time adaptation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1414–1423, 2024.

[53] Guy Gur-Ari, Daniel A Roberts, and Ethan Dyer. Gradient descent happens in a tiny subspace. *arXiv preprint arXiv:1812.04754*, 2018.

[54] Can Yaras, Peng Wang, Wei Hu, Zhihui Zhu, Laura Balzano, and Qing Qu. The law of parsimony in gradient descent for learning deep linear networks. *arXiv preprint arXiv:2306.01154*, 2023.

[55] Tao Li, Lei Tan, Qinghua Tao, Yipeng Liu, and Xiaolin Huang. Low dimensional landscape hypothesis is true: Dnns can be trained in tiny subspaces. *arXiv preprint arXiv:2103.11154*, 2021.

[56] Martin Gauch, Maximilian Beck, Thomas Adler, Dmytro Kotsur, Stefan Fiel, Hamid Eghbalzadeh, Johannes Brandstetter, Johannes Kofler, Markus Holzleitner, Werner Zellinger, et al. Few-shot learning by dimensionality reduction in gradient space. In *Conference on Lifelong Learning Agents*, pages 1043–1064. PMLR, 2022.

[57] Jinlong Liu, Yunzhi Bai, Guoqing Jiang, Ting Chen, and Huayan Wang. Understanding why neural networks generalize well through gsnr of parameters. In *International Conference on Learning Representations*, 2020.

[58] Ziwei Ji and Matus Telgarsky. Directional convergence and alignment in deep learning. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 17176–17186. Curran Associates, Inc., 2020.

[59] Karthik Abinav Sankararaman, Soham De, Zheng Xu, W Ronny Huang, and Tom Goldstein. The impact of neural network overparameterization on gradient confusion and stochastic gradient descent. In *International conference on machine learning*, pages 8469–8479. PMLR, 2020.

[60] Maksym Andriushchenko and Nicolas Flammarion. Understanding and improving fast adversarial training. *Advances in Neural Information Processing Systems*, 33:16048–16059, 2020.

[61] Zhiqiang Gao, Shufei Zhang, Kaizhu Huang, Qiufeng Wang, and Chaoliang Zhong. Gradient distribution alignment certificates better adversarial domain adaptation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 8937–8946, 2021.

[62] Luis Barba, Martin Jaggi, and Yatin Dandi. Implicit gradient alignment in distributed and federated learning. In *AAAI Conference on Artificial Intelligence, AAAI*, volume 22, 2021.

[63] Stanislav Fort, Paweł Krzysztof Nowak, Stanislaw Jastrzebski, and Srini Narayanan. Stiffness: A new perspective on generalization in neural networks. *arXiv preprint arXiv:1901.09491*, 2019.

[64] Gunshi Gupta, Karmesh Yadav, and Liam Paull. Look-ahead meta learning for continual learning. *Advances in Neural Information Processing Systems*, 33:11588–11598, 2020.

[65] Yichen Wu, Hong Wang, Peilin Zhao, Yefeng Zheng, Ying Wei, and Long-Kai Huang. Mitigating catastrophic forgetting in online continual learning by modeling previous task interrelations via pareto optimization. In *Forty-first International Conference on Machine Learning*, 2024.

[66] Yichen Wu, Hong Wang, Long-Kai Huang, Yefeng Zheng, Peilin Zhao, and Ying Wei. Enhanced gradient aligned continual learning via pareto optimization. In *International Conference on Learning Representations*, 2024.

[67] Mateusz Michalkiewicz, Masoud Faraki, Xiang Yu, Manmohan Chandraker, and Mahsa Baktashmotlagh. Domain generalization guided by gradient signal to noise ratio of parameters. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6177–6188, October 2023.

[68] Giambattista Parascandolo, Alexander Neitz, Antonio Orvieto, Luigi Gresele, and Bernhard Schölkopf. Learning explanations that are hard to vary. In *9th International Conference on Learning Representations, ICLR*, 2021.

[69] Yuge Shi, Jeffrey Seely, Philip H. S. Torr, Siddharth Narayanaswamy, Awni Y. Hannun, Nicolas Usunier, and Gabriel Synnaeve. Gradient matching for domain generalization. In *The Tenth International Conference on Learning Representations, ICLR*, 2022.

[70] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. *Advances in Neural Information Processing Systems*, 33:5824–5836, 2020.

[71] Yunshu Du, Wojciech M Czarnecki, Siddhant M Jayakumar, Mehrdad Farajtabar, Razvan Pascanu, and Balaji Lakshminarayanan. Adapting auxiliary losses using gradient similarity. *arXiv preprint arXiv:1812.02224*, 2018.

[72] Mihai Suteu and Yike Guo. Regularizing deep multi-task networks using orthogonal gradients. *arXiv preprint arXiv:1912.06844*, 2019.

[73] Maren Mahsereci, Lukas Balles, Christoph Lassner, and Philipp Hennig. Early stopping without a validation set. *arXiv preprint arXiv:1703.09580*, 2017.

[74] Chirag Agarwal, Daniel D'souza, and Sara Hooker. Estimating example difficulty using variance of gradients. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10368–10378, 2022.

[75] Mahsa Forouzesh and Patrick Thiran. Disparity between batches as a signal for early stopping. In *Machine Learning and Knowledge Discovery in Databases. Research Track: European Conference, ECML PKDD 2021, Bilbao, Spain, September 13–17, 2021, Proceedings, Part II 21*, pages 217–232. Springer, 2021.

[76] Suqin Yuan, Lei Feng, and Tongliang Liu. Early stopping against label noise without validation data. In *International Conference on Learning Representations*, 2024.

[77] David Bonet, Antonio Ortega, Javier Ruiz-Hidalgo, and Sarath Shekkizhar. Channel-wise early stopping without a validation set via nnk polytope interpolation. In *2021 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pages 351–358. IEEE, 2021.

[78] Yingbin Bai, Erkun Yang, Bo Han, Yanhua Yang, Jiatong Li, Yinian Mao, Gang Niu, and Tongliang Liu. Understanding and improving early stopping for learning with noisy labels. *Advances in Neural Information Processing Systems*, 34:24392–24403, 2021.

[79] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88: 303–338, 2010.

[80] Bryan C Russell, Antonio Torralba, Kevin P Murphy, and William T Freeman. Labelme: a database and web-based tool for image annotation. *International journal of computer vision*, 77:157–173, 2008.

[81] Li Fei-Fei, Rob Fergus, and Pietro Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In *2004 conference on computer vision and pattern recognition workshop*, pages 178–178. IEEE, 2004.

[82] Myung Jin Choi, Joseph J Lim, Antonio Torralba, and Alan S Willsky. Exploiting hierarchical context on a large database of object categories. In *2010 IEEE computer society conference on computer vision and pattern recognition*, pages 129–136. IEEE, 2010.

[83] Shuai Wang, Daoan Zhang, Zipei Yan, Jianguo Zhang, and Rui Li. Feature alignment and uniformity for test time adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20050–20060, 2023.

[84] Eungyeup Kim, Mingjie Sun, Aditi Raghunathan, and Zico Kolter. Reliable test-time adaptation via agreement-on-the-line. *arXiv preprint arXiv:2310.04941*, 2023.

[85] Malik Boudiaf, Romain Mueller, Ismail Ben Ayed, and Luca Bertinetto. Parameter-free online test-time adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8344–8353, 2022.

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

[87] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. *Toronto, ON, Canada*, 2009.

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

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

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

# Supplementary Materials for "A Layer Selection Approach to Test Time Adaptation"

In the supplementary section, we provide a comprehensive discussion of the experimental setups and the proposed approach, GALA. The supplementary material is organized as follows:

- **Sec. A.1**: Experimental results on Continual TTA benchmark.
- **Sec. A.2**: Analysis of GALA.
- **Sec. A.3**: Related work section.
- **Sec. A.4**: Implementation details of the Domainbed benchmark (for Tab. 1, and Fig. 3).
- **Sec. A.5**: Implementation details of the Continual TTA benchmark (for Tab. 3).
- **Sec. A.6**: Implementation details of the Tiny Domainbed benchmark (for Tab. 2), including the rationale for selecting critical shifts from the Domainbed benchmark.
- **Sec. A.7**: An in-depth discussion of GALA, including pseudocode and an analysis of its balance between gradient alignment and magnitude.
- **Sec. A.8**: Concluding discussion section.

The numbering of figures, tables, and equations in this supplementary material continues from the main paper to ensure consistency and avoid repetition.

## A.1 Continual TTA results

We follow the evaluation protocol as described in Wang et al. [13], evaluating performance on two datasets-backbones: 1) CIFAR10C [14] with WideResNet-28 [15] and CIFAR100C [14] with ResNeXt-29 [16]. The pretrained models are trained as described in Robustbench [17]. Mean and standard deviation are reported across the 15 corruption types. Further details are given in Appendix A.5.

Table 3: Accuracy (%) of layer selection methods on Continual TTA benchmark (with the setup described in Sec. A.1). The best method for a given TTA loss is in bold.

| TTA | Method | CIFAR10C ↓ | CIFAR100C ↓ |
|---|---|---|---|
| ERM | | 43.50 (±18.7) | 46.40 (±15.7) |
| PL | All layers | 88.72 (±1.2) | 98.63 (±1.5) |
| | GALA | **28.68** (±**6.6**) | **33.69** (±**5.7**) |
| SHOT | All layers | 89.33 (±2.3) | 97.32 (±4.8) |
| | GALA | **20.46** (±**7.7**) | **32.87** (±**5.6**) |

The key takeaways based on the results from Tab. 3 are:

- Performance degradation by training all layers is worse in the Continual TTA benchmark containing multi-domain shifts than degradation in the Domainbed benchmark containing single-domain shifts. Moreover, more severe degradations are observed in CIFAR100C, which has 100 classes, compared to CIFAR10, which includes 10 classes, despite similar ERM performance on both datasets.
- GALA consistently outperforms ERM by about 15% and all layers TTA baseline by about 65%, despite severe degradation.

## A.2 Analysis of GALA

In this section, we evaluate the impact of different design choices and hyperparameters of GALA in Tab. 4, supporting choices presented in Tab. 1. For the partitioning setting, *Single block* means a single block of many layers is updated at each iteration, *Single layer* corresponds to the best layer selected for the update, and *Multiple layers* corresponds to individually best layers selected for the update based on the cosine distance and the threshold. A layer denotes either a convolution or batch norm layer. Also, a window size of $\infty$ implies no reset. Some important observations stemming from Tab. 4:

- Layer granularity performs better than block granularity. At layer granularity, GALA has better fine-grained control over choosing the layers to adapt, improving performances in all cases tested.

Table 4: Accuracy (%) under different experimental conditions. The values are averaged for each backbone and TTA loss of the Domainbed benchmark.

| Setting | Condition | Resnet-18 ↑ PL | Resnet-18 ↑ SHOT | Resnet-50↑ PL | Resnet-50↑ SHOT |
|---|---|---|---|---|---|
| **Partitioning** | Single block | 66.19 | 65.8 | 70.21 | 68.37 |
| | Single layer | 66.31 | 66.78 | 71.07 | 70.13 |
| | Multiple layers | 63.13 | 65.36 | 69.11 | 68.29 |
| **Threshold** | 0.5 | 66.31 | 66.78 | 71.07 | 70.13 |
| | 0.75 | 66.31 | 66.78 | 71.07 | 70.13 |
| | 0.99 | 66.24 | 66.96 | 71.07 | 70.13 |
| **Window Size** | 5 | 66.37 | 66.61 | 70.87 | 70.01 |
| | 20 | 66.31 | 66.78 | 71.07 | 70.13 |
| | $\infty$ | 65.71 | 66.59 | 70.76 | 70.42 |
| **Batch Size = 1** | All Layers | 37.47 | 29.67 | 33.93 | 32.81 |
| | GALA | 64.12 | 64.29 | 70.45 | 70.24 |

Table 5: Accuracy (%) under different experimental conditions. The values are averaged for each dataset and TTA loss of the Continual TTA benchmark.

| Setting | Condition | CIFAR10C ↑ PL | CIFAR10C ↑ SHOT | CIFAR100C ↑ PL | CIFAR100C ↑ SHOT |
|---|---|---|---|---|---|
| **Continual TTA** | No Reset | 73.6 | 76.41 | 64.75 | 64.87 |
| | With Reset | 71.32 | 79.54 | 66.31 | 67.13 |

- Adaptation with the best single layer is much better than with the best multiple layers. Cosine distance can correctly identify the single best layer to train, although it may still struggle to determine the best set of multiple layers to update.
- Optimal reset-window size can improve performance. We see that a reset window size of 20 works reasonably well across the backbones and the TTA losses tested on Domainbed.
- The choice of selection threshold is not very sensitive. A threshold of 0.75 seems to work across the board without being too restrictive.

In the following section, we briefly analyze some aspects of the proposed approach.

**Proposed cosine distance criterion effectively balances gradient magnitude and direction.** Let us first rewrite the GALA criterion in Eq. 4 for a given layer $l$ in terms of $T = \|\mathbf{TD}_{i-1,l}\|$, $u = \|\mathbf{u}_{i,l}\|$ and the angle $\beta$ between $\mathbf{TD}_{i-1,l}$ and $\mathbf{u}_{i,l}$. Using the Pythagorean theorem, we obtain:

$$\cos(\alpha) = \frac{T\cos(\beta) + u}{\sqrt{(T + u\cos(\beta))^2 + (u\sin(\beta))^2}}. \tag{7}$$

We observe that our criterion depends on the norm $T$ of the total displacement, the norm $u$ of the update, and their alignment, given by the angle $\beta$ between these vectors. Fig. 4 shows the cosine metric plots. We see that while alignment is crucial for large displacements, the update's magnitude can also dominate for small displacements. For example, consider two layers with the same norm $T$ but different updates $\mathbf{u}_1$ and $\mathbf{u}_2$. If $\|\mathbf{u}_1\|$ is smaller than $\|\mathbf{u}_2\|$ but $\mathbf{u}_1$ is more aligned with its displacement, two scenarios arise:

1. For larger $T$, GALA selects layer 1, favoring the alignment and exploiting the learned direction. This scenario would seem more common during TTA.
2. For small $T$, GALA selects layer 2, favoring the magnitude, and can explore over different directions. This can occur for initial samples.

Consequently, GALA effectively balances the gradient magnitude and the direction of gradients for selecting the best layer. More discussion is in Appendix A.7.

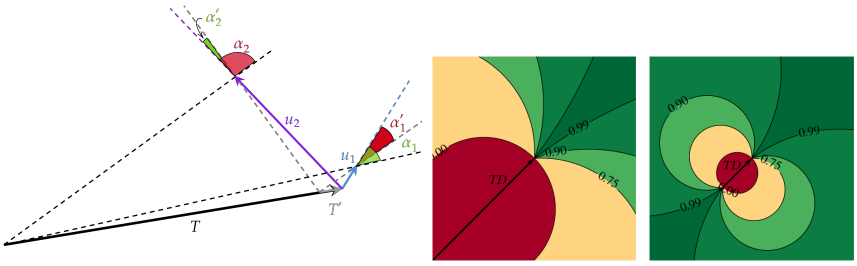
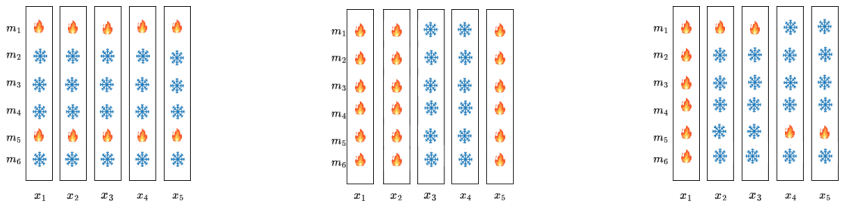



Figure 4: Effect of magnitude of $u$ on cosine distance criterion. **Left:** Consider two vectors such that $u_1$ is smaller than $u_2$ but is better aligned with its displacement. For large displacements ($T$), alignment becomes crucial and GALA selects $u_2$. For small displacements ($T^{'}$), the update's magnitude can dominate the criterion, and GALA selects $u_1$. **Middle and Right:** Plot of cosine metric values with level curves. Alignment prevails for small updates compared to the total displacement (Middle). But, for updates with large magnitude compared to total displacement (Right), large cosine values can be obtained even for misaligned updates.




(a) Layer-wise (b) Sample-wise (c) GALA

Figure 5: Different adaptation strategies: (a) TTA approaches typically adapt a fixed set of layers for all the samples. (b) Sample selection-based TTA approaches skip the adaptation of all layers on a few unreliable samples. (c) GALA is more flexible and can dynamically control the adaptation of individual layers per sample.

**Proposed layer selection framework offers a more flexible adaptation strategy for TTA.** The selection of layers in existing TTA approaches typically remains unchanged across different shifts. On the other hand, sample selection-based TTA [18] approaches aim to improve performance by skipping the adaptation of all layers on a few unreliable samples. Based on Eq. 2 and Fig. 5, we can see that GALA is more flexible and general than the existing layer selection and sample selection strategies in TTA for performing layerwise adaptation.

**Reset mechanism seems beneficial in multi-domain shift settings.** Comparing GALA with and without reset on Tab. 5, we see that while reset yields only marginal improvement on Domainbed, a single-domain shift benchmark, its benefits are more evident on a multi-shift benchmark like Continual TTA. This indicates that the reset mechanism's ability to facilitate slight adjustments in the overall gradient update direction may be advantageous in a continuously changing testing domain.

**GALA is quite robust on single sample adaptation.** In Tab. 4, we show that in the adverse setting of batch size of 1, while existing TTA approaches witness severe performance degradation, GALA improves on *all layers* baseline on Domainbed.

## A.3 Related Work

**Regularization in test time adaptation** addresses performance degradation. Recent works have focused on improving pseudolabel quality [13, 19], enhancing class prototypes [5, 20, 21], and constraining model behavior through domain alignment [22–24] and loss regularization [18, 25–28]. In contrast, our work introduces a parameter-centric approach to regularize TTA, which we argue can be more effective for efficient adaptation while allowing it to be combined with existing techniques, given its complementarity.

**Layer selection** is known for enhancing feature sharing in multi-task learning [29–32] or continual learning [33–35], and also explaining the learned features [36, 37]. It can also be helpful for out-of-distribution detection [38–41] and creation of robust pretrained models for self-supervised learning [42–44] and domain generalization [45]. Effective parameter selection also significantly impacts fine-tuning [4, 46–48] and learning of non-spurious features [49, 50]. While some studies [4, 51, 52] have explored parameter impacts on specific TTA losses, their impact is somewhat limited and potentially inapplicable to other TTA losses. In contrast, we propose a gradient-aligned layer adaptation framework for TTA, offering greater flexibility than existing strategies and demonstrating improved performance across various TTA losses.

**Gradient alignment studies**. Gur-Ari et al. [53] discuss the phenomenon of gradient descent occurring within a tiny subspace. Yaras et al. [54] demonstrate the presence of a low-dimensional structure in learning dynamics. Li et al. [55] reveal that neural networks can be effectively trained in lower-dimensional subspaces. Gauch et al. [56] show that gradient descent within a tiny subspace enhances generalization in few-shot learning. Liu et al. [57] indicate that a high gradient signal-to-noise ratio can lead to improved generalization in neural networks. Ji and Telgarsky [58] find that the gradients of neural networks converge to a single direction. Sankararaman et al. [59] show that gradient alignment accelerates the training speed of neural networks. Building upon these approaches, this paper proposes a novel cosine distance-based criterion for layer selection in the context of test-time adaptation.

**Applications of gradient alignment**. Andriushchenko and Flammarion [60] and Gao et al. [61] propose using gradient alignment as a regularizer for adversarial training. Barba et al. [62] apply gradient alignment within the context of federated learning. Fort et al. [63] demonstrate that gradient alignment can be useful for detecting overfitting. Gupta et al. [64], along with Wu et al. [65, 66], show that gradient alignment can mitigate catastrophic forgetting in continual learning. Michalkiewicz et al. [67], Parascandolo et al. [68] and Shi et al. [69] utilize gradient alignment for domain generalization. Yu et al. [70] show that gradient alignment aids in multi-task learning, a finding supported by Du et al. [71] and Suteu and Guo [72]. Building on these approaches, we propose a novel formulation of gradient alignment for an online and unsupervised application of test time adaptation.

**Gradient alignment-based early stopping**. Recent works have proposed various approaches or criteria for early stopping without a validation set. Mahsereci et al. [73] introduced an evidence-based criterion based on the variance of gradients [74]. Forouzesh and Thiran [75] proposed using gradient disparity across samples as a criterion. Yuan et al. [76] suggested performing early stopping by tracking the model's predictions on the samples. Most of these existing works have demonstrated the effectiveness of their proposed approaches, often in the context of multi-epoch and supervised learning [77] or noisy learning [78]. Similar to these approaches, our novel cosine distance-based criterion can be used to perform layer-wise early stopping without a validation set, preventing degradation in test-time adaptation and beyond.

## A.4 Experimental Details of Domainbed

### A.4.1 Dataset Details

Domainbed [2] consists of four domain generalization datasets:

- PACS [8] consists of different object images from four domains: Art, Cartoon, Photo, and Sketch. It comprises of 9,991 samples across 7 class labels (i.e., dog, elephant, giraffe, guitar, horse, house, and person).
- VLCS [6] consists of photographic images from four domains/datasets: PASCAL VOC207 [79], LabelMe [80], Caltech 101 [81], and SUN09[82]. It comprises of 10,729 samples across 5 class labels (i.e., bird, car, chair, dog, and person).
- TerraIncognita [7] consists of images of wild animals taken at different locations, which make up the four domains: L100, L38, L43, and L46. It comprises of 24,788 samples across 10 class labels (i.e., bird, bobcat, cat, coyote, dog, empty/no animal, opossum, rabbit, raccoon, squirrel).
- Office-Home [9] consists of different object images typically seen in offices and homes from four domains: Art, Clipart, Product, and Real World. It comprises of 15,588 samples across 65 class labels (e.g., bottle, computer, hammer, pen).

### A.4.2 Evaluation Details

We follow the evaluation protocol as described in Iwasawa and Matsuo [5]. In Domainbed, the pretrained model is trained on all but one domain. All the domains on which the pretrained model is trained are referred to as training domains, and the remaining domain is referred to as the testing domain. We follow the dataset splits used in T3A [5]. Each domain is split into a big and a small split. Specifically, the domains are split into 80% and 20%. The big split of training domains is used for training the pretrained model and is referred to as training splits. The small splits of training domains are referred to as validation splits. The big split of the testing domain is used to evaluate the domain and is referred to as the testing split. This is the split where test time adaptation is performed for each minibatch before inference. We consider three seeds for the results in Tab. 1 and one seed in Tab. 4, Fig. 4, and Fig. 6. Each seed generates a new training and testing split from training and testing domains. See Figure 1 "Data configuration for a benchmark with four domains" in Domainbed [2] supplementary.

The performance on each domain or shift is obtained by averaging across multiple seeds. Next, we obtain the performance on each dataset by averaging across all domains in the dataset. Finally, we get performance on Domainbed by averaging across all the datasets in Domainbed.

### A.4.3 Hyperparameters and Model Selection

#### A.4.3.1 Pretrained Model

We follow the pretraining protocol as described in Iwasawa and Matsuo [5]. We use ERM for pretraining the model similar to other works in TTA [5, 20, 83]. We consider two backbones: Resnet-18 and Resnet-50, with batch normalization layers. The backbone networks are trained using ERM and Adam optimizer with a batch size of 32. We follow the training-domain validation-based model selection [2, 5] where we choose the hyperparameters that maximize the accuracy of the pretrained model on the validation splits. Please refer to Domainbed [2] and T3A [5] for a detailed discussion on hyperparameters and the range used.

#### A.4.3.2 TTA Approaches

We follow the adaptation protocol as described in Iwasawa and Matsuo [5]. Similar to other works in TTA [12, 84–86], our hyperparameter tuning protocol of TTA methods is based on Zhao et al. [12], where we choose the best hyperparameter set for each TTA method under consideration. We consider two popular TTA methods from Iwasawa and Matsuo [5]: pseudo-labeling (PL) [10] and SHOT [11]. Please refer to Iwasawa and Matsuo [5] for a detailed discussion on hyperparameters and the range used.

#### A.4.3.3 Baseline Layer Selection Methods

We perform a model selection for each baseline as described in the original implementations. ERM [2, 5], *All Layers* [5], and AutoRGN [4] baselines do not have any hyperparameters. The hyperparameter for AutoSNR [4] baseline is tuned as described in Lee et al. [4].

## A.5 Experimental Details of Continual TTA

### A.5.1 Dataset Details

Continual TTA benchmark [13] consists of two datasets, CIFAR10-C and CIFAR100-C, widely used for evaluating the robustness of classification networks under various corruptions, particularly in the context of test-time adaptation (TTA). These datasets are derived from the original CIFAR10 and CIFAR100 [87], which contain 50,000 training and 10,000 test images across 10 and 100 categories, respectively. In CIFAR10-C and CIFAR100-C [88], 15 types of corruptions, each with 5 levels of severity, are applied to the test images of their clean counterparts. This results in 10,000 corrupted images for each corruption type in both the datasets.

### A.5.2 Evaluation Details

We follow the evaluation protocol as described in Wang et al. [13]. We utilize a model pre-trained on the clean training set of the CIFAR10 or CIFAR100 dataset. During test time, corrupted images are provided to the network in an online fashion. We continually adapt the source pretrained model

to each corruption type sequentially without resetting to the pretrained model. The CIFAR10 and CIFAR100 experiments follow this online continual test-time adaptation scheme, with evaluations conducted under the highest corruption severity level 5. The evaluation is based on the online prediction results immediately after encountering the data.

### A.5.3 Hyperparameters and Model Selection

**Pretrained Model** We follow the pretraining protocol as described in Wang et al. [13]. For our experiments on CIFAR10C and CIFAR100C, we utilize pre-trained models from the RobustBench benchmark [17] similar to previous works in test time adaptation [13, 18, 89, 90]. Specifically, for CIFAR10C, we employ a WideResNet-28 [15] model, and for CIFAR100C, we adopt a pre-trained ResNeXt-29 [16] model, which is one of the default architectures for CIFAR100 in RobustBench.

**TTA Approaches** We follow the evaluation protocol as described in Wang et al. [13]. We update the model with one gradient step per test point at each iteration, utilizing the Adam optimizer with a learning rate of 1e-3. The hyperparameters employed are consistent with those recommended by Wang et al. [13]. To facilitate comparison with Domainbed benchmark results, we incorporate the same two test-time adaptation methods: pseudo-labeling (PL) [10] and SHOT [11].

**Baseline Layer Selection Methods** We perform a model selection for each baseline as described in the original implementations. ERM [13, 17] and *All Layers* [5] baselines do not have any hyperparameters.

## A.6 Experimental Details of Tiny Domainbed

### A.6.1 Dataset and Shift Details

We create Tiny-Domainbed from Domainbed by selecting the following critical shifts:

- Three shifts from the Terra Incognita benchmark: L100, L38, and L43;
- Two shifts from the PACS benchmark: Cartoon and Sketch;
- One shift from the VLCS benchmark: SUN09.

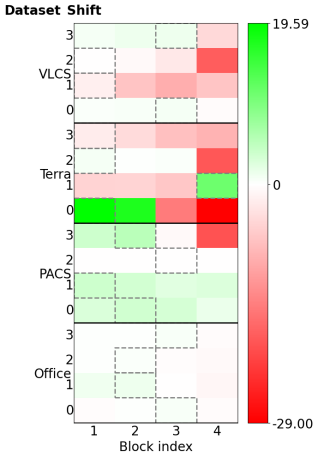
### A.6.2 Discussion on Chosen Shifts

Creating Tiny-Domainbed aims to make the smallest possible setup of Domainbed, which contains all the challenging shifts or domains in Domainbed while being computationally light for ease of analysis and comparison. To identify the critical shifts of Domainbed, we refer to the heatmap of performance improvement of blocks vs. shifts in Domainbed in Fig. 6. We refer to blocks and layers interchangeably in this section.

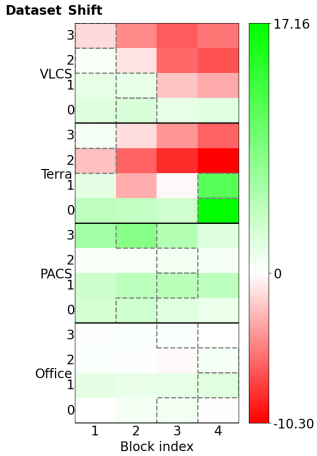
Based on heatmaps of the Resnet-18 backbone for pseudolabelling and SHOT loss-based TTA methods in Fig. 6, we identify the critical shifts in Domainbed which satisfy the following two important properties:

- **Property 1: Shifts with brightest green/red blocks.** A bright green layer implies that adapting this layer improves performance over ERM. Similarly, a bright red layer implies that adapting this layer can degrade the performance with respect to ERM. The shifts with the brightest green or red layers are important because any layer selection criterion must do well on these shifts. The inability to choose bright green layers while adapting to these shifts is a missed opportunity for performance improvement of the layer selection approach. Again, being unable to avoid bright red layers in these shifts can result in significant performance degradation. In Fig. 6, we see that the following shifts on the Resnet-18 backbone have the brightest green/red layers: $env0, env1, env3$ in PACS; $env2, env3$ in VLCS; $env0, env1, env2, env3$ in Terra Incognita; *none* in OfficeHome.
- **Property 2: Shifts whose best block changes with the TTA loss function.** We make a striking observation that for certain shifts in Domainbed, the best block for a given shift can depend on the TTA loss used. This implies that the layer selection criterion must consider TTA loss to choose the best layers to adapt. In Fig. 6, we see that the following shifts on the Resnet-18 backbone experience a change in the location of the best block due to a change in the TTA method: $env1$ in PACS; $env0, env3$ in VLCS; $env0$ in Terra Incognita; $env1, env2$ in OfficeHome.

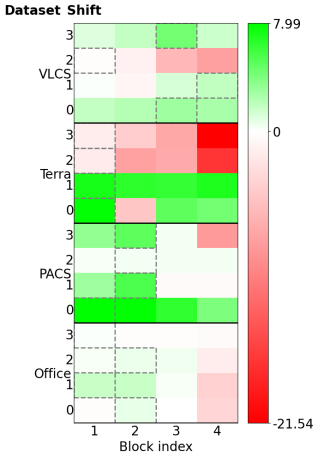
To create the Tiny-Domainbed benchmark, we select the most critical shifts with the brightest green/red blocks, and the location of the best block changes with the TTA loss function. Based on this, we identify the following shifts from Domainbed to be included in the Tiny-Domainbed benchmark:



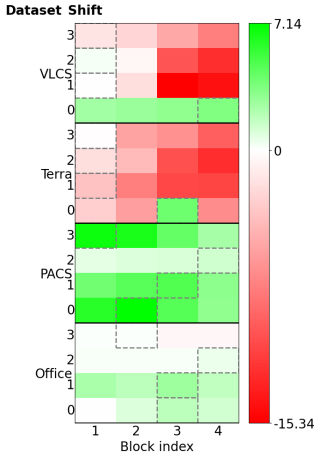

(a) Resnet-18 with PL loss




(b) Resnet-18 with SHOT loss




(c) Resnet-50 with PL loss




(d) Resnet-50 with SHOT loss

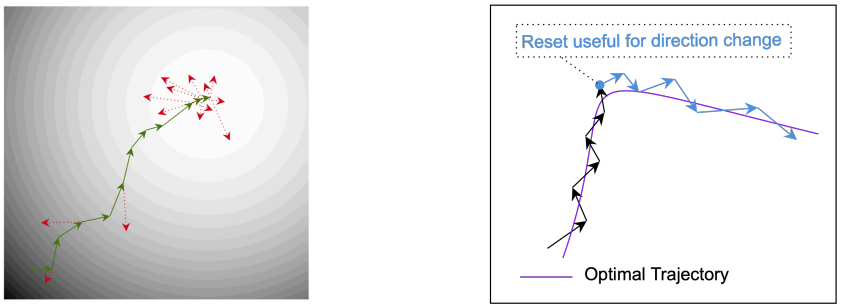
Figure 6: Heatmap of Performance improvement (%) per-block on Domainbed benchmark for Resnet-18 (same as Figure 4) and Resnet-50. Performance improvement is the difference between the TTA accuracy of a given block/layer and ERM accuracy for the same shift. Positive performance improvements are shown in green, and negative performance improvements (or degradation) are in red. Using the bounding box, we highlight the best block per loss and dataset shift.

- Shifts that satisfy both the properties: $env3$ in PACS; $env3$ in VLCS; $env0$ in Terra Incognita.
- Shifts that only satisfy property 1 but are included in Tiny Domainbed: $env1$ in PACS; $env1, env2$ in Terra Incognita.

This gives us our final list of critical shifts from Domainbed included in the Tiny-Domainbed benchmark: three shifts from the Terra Incognita benchmark: L100, L38, and L43; two shifts from the PACS benchmark: Cartoon and Sketch, and one shift from the VLCS benchmark: SUN09.

### A.6.3 Evaluation Details

We follow the evaluation protocol as described in Iwasawa and Matsuo [5] and is described in detail in Sec. A.4.2. We obtain the performance on a given testing domain similar to the evaluation protocol of Domainbed. However, we obtain the final performance on Tiny-Domainbed by averaging across only the selected domains or shifts (identified as critical shifts in Sec. A.6.1) on the Resnet-18 backbone and two TTA losses.




(a) Gradient alignment (b) Helpful reset

Figure 7: Intuition for proposed approaches: **(a)** As the model reaches closer to minima, the individual sample gradients start to be misaligned with gradients of previous samples [73–75]. We leverage this misalignment to identify trainable layers. **(b)** While effective in moving in the direction of most aligned gradients, the introduced criterion based on angular deviation could prevent adaptation when a direction change is needed, even if the following updates (or gradients) are aligned. A reset of the past horizon (i.e., gradients of previous samples) considered in the alignment condition can help resolve such situations.

**Evaluation Metrics**. We will explain the various metrics employed to compare different block or layer selection methods, as used in Table 3:

- **TTA Accuracy**: We abbreviate it as *TTA acc*. It refers to the accuracy of testing samples from the target domain observed during adaptation. This metric follows the same evaluation protocol as in Tables 1 and 2 to generate the TTA results, providing insight into the performance of different layer selection approaches at test time.
- **Generalization**: It measures the accuracy on the held-out split of the target domain after the model has completed adaptation on all the samples from the same target domain. This metric indicates how each layer selection method aids in adapting the model to the target domain.
- **Forgetting**: This metric quantifies the drop in accuracy on the held-out split of source domains after the pretrained model has adapted to all samples of the target domain, which differs from the source domains on which the pretrained model was initially trained. This metric helps us assess the degree of forgetting of source features due to various layer selection methods.
- **Rank Correlation**: In this metric, we measure the Spearman correlation of layer selection ranks between the oracle and proposed layer selection methods. Oracle TTA performance, as depicted in Figure 6, ranks the four blocks for each configuration. Similarly, different layer selection methods adapt a layer with a specific frequency during TTA, resulting in a ranking of the four blocks. We evaluate the relationship between these two ranking methods using Spearman rank correlation. This correlation provides insight into how well the proposed layer selection methods' ranking of layers aligns with the oracle ranking on Tiny-Domainbed.

### A.6.4 Hyperparameters and Model Selection

We follow the model selection and hyperparameter tuning protocol for pretrained models and TTA approaches as described in Iwasawa and Matsuo [5] and is described in detail Sec. A.4.3. We consider only the Resnet-18 backbone with the batch normalization layers for ease of analysis. We consider two TTA losses: pseudolabelling and SHOT. We use *Block* granularity-based layer selection and report results for a single seed for easy analysis. Please note that although we perform block-based layer selection in Tiny-Domainbed, we interchangeably refer to block selection or layer selection in this section.

**Layer Selection Methods**. In Tab. 2, we compare the GALA method with the following oracle (*Best Block* and *Worst Block*) and baseline methods (All Blocks and Random Block):

- **All Blocks**: This is analogous to the *All Layers* baseline in Tables 1 and 2, where all blocks of the model are adapted at each adaptation step.
- **Random Block**: During each adaptation step for a given sample, a block is chosen at random and adapted accordingly.

**Algorithm 1** Gradient-Aligned Layer Adaptation (GALA)

**Initialization:** Pretrained model: $f_{\theta_0}(x)$, Anchor model parameters: $\theta_{\text{anchor}} = \theta_0$, Adaptation method: `TTA`, Window size: $s$, Mask threshold: $\lambda$
**Input for step $i$:** Sample $x_i$, anchor $\theta_{\text{anchor}}$ and current model $\theta_{i-1}$
$\mathbf{u}_i = \texttt{TTA}(x_i, \theta_{i-1})$
**for** each layer $l$ **do**
    $\mathbf{TD}_{i-1,l} = \theta_{i-1,l} - \theta_{\text{anchor},l}$
    $\cos(\alpha_{i,l}) = \frac{\mathbf{u}_{i,l} \cdot (\mathbf{u}_{i,l} + \mathbf{TD}_{i-1,l})}{\|\mathbf{u}_{i,l}\|_2 \, \|\mathbf{u}_{i,l} + \mathbf{TD}_{i-1,l}\|_2}$
    $m_{i,l} = \begin{cases} 1 & \text{if } \cos(\alpha_{i,l}) > \lambda \\ 0 & \text{otherwise} \end{cases}$
    $\theta_{i,l} = \theta_{i-1,l} + m_{i,l}\mathbf{u}_{i,l}$
**end for**
$r = \lfloor (i-1)/s \rfloor$
**if** $i == r * s$ **then**
    $\theta_{\text{anchor}} = \theta_i$
**end if**
**Output at step $i$:** Prediction $f_{\theta_i}(x_i)$, Updated model $f_{\theta_i}$

- **Best Block**: In this oracle layer selection method, the best-performing block for each shift and loss function, as identified in Figure 6, is adapted for all adaptation steps of the model.
- **Worst Block**: This oracle method adapts the worst-performing block for each shift and loss function, as identified in Figure 6, for all adaptation steps of the model.

## A.7 Discussion on GALA

Intuition for proposed approaches is given in Fig. 7.

### A.7.1 Pseudocode

A pseudocode of GALA is given in Algorithm 1.

### A.7.2 Implementation details of GALA

GALA has two hyperparameters, namely *window size* and *mask threshold*. In Appendix A.2, we show that GALA is not overly sensitive to its hyperparameters and, therefore, use a fixed value (*window size* = 20 and *mask threshold*=0.75) across all setups when comparing with the baselines. Total displacement computed over the initial few samples may be unreliable, which can result in incorrect layers selected for adaptation. We address this by scaling the masked updates for an initial few samples in the reset window. One could also use an earlier anchor model from the previous reset window, but scaling the mask for a few initial minibatches seems to suffice.

Based on Appendix A.2, we note that GALA performs the best when the pretrained model is adapted only with the best single layer identified by GALA (the one with the largest cosine distance above the selection threshold) but not with multiple or the top $k$-best layers identified by GALA (i.e., layers whose cosine distance is larger than the selection threshold). Therefore, we use GALA with Single-layer partitioning across all setups when comparing with the baselines. An important point to note is that GALA adapts all the layers for the first sample in a reset window. It adapts the most gradient-aligned layer per sample for all the other samples. (However, in the Tiny-Domainbed benchmark, we use Single-block partitioning since the analysis is performed at Block granularity.)

Since GALA performs the best with Single-layer partitioning instead of Multi-layer partitioning, it implies that GALA can often identify a good layer to adapt but may not identify all the top $k$-best layers to adapt above the selection threshold. This can be viewed as one of the limitations of GALA and can potentially be a fruitful research direction for future works. We note that one can address this limitation by tuning the hyperparameters of GALA (especially *mask threshold*) in each setup, similar to the recommendation made by Zhao et al. [12]. However, we avoid any hyperparameter tuning of GALA in the paper and show that adapting the model with the best single-layer identified by GALA can outperform existing baselines.

### A.7.3 Relationship between GALA and Eq. 7

For notational simplicity, we rewrite the following terms

- $\mathbf{u} = \mathbf{u}_{i,l}$, the current sample's update for layer $l$.
- $u = \|\mathbf{u}\|_2$, the magnitude of the current sample's update.
- $\mathbf{T} = \mathbf{TD}_{i-1,l}$, the total displacement undergone in previous steps by layer $l$.
- $T = \|\mathbf{T}\|_2$, the magnitude of the total displacement. This is the *magnitude* used in Sec. A.2 and Sec. A.7.4.
- $\beta = \beta_{i,l}$ is the angle between $\mathbf{u}$ and $\mathbf{T}$.
- $\cos(\beta)$ is the alignment of $\mathbf{u}$ with $\mathbf{T}$. This is the *alignment* used in Sec. A.2 and Sec. A.7.4. We also interchangeably refer to it as *direction* since $\beta$ is the angle $\mathbf{u}$ makes with $\mathbf{T}$.
- $\alpha = \alpha_{i,l}$ is the angle between $\mathbf{u}$ and $\mathbf{u} + \mathbf{T}$.
- $\cos(\alpha)$ is the proposed criterion of GALA. In Sec. A.2 and Sec. A.7.4, we note that the proposed cosine distance criterion effectively balances *magnitude* and *alignment*.

Based on the above definitions, the *alignment* is given by

$$\cos(\beta) = \frac{\mathbf{u} \cdot \mathbf{T}}{u\, T} \tag{8}$$

Let us begin by expanding the numerator of Eq. 4,

$$\cos(\alpha) = \frac{\mathbf{u} \cdot (\mathbf{T} + \mathbf{u})}{u\|\mathbf{T} + \mathbf{u}\|_2} \tag{9}$$

$$= \frac{\mathbf{u} \cdot \mathbf{T} + \mathbf{u} \cdot \mathbf{u}}{u\|\mathbf{T} + \mathbf{u}\|_2} \tag{10}$$

$$= \frac{uT\cos(\beta) + u^2}{u\|\mathbf{T} + \mathbf{u}\|_2} \tag{11}$$

$$= \frac{T\cos(\beta) + u}{\|\mathbf{T} + \mathbf{u}\|_2} \tag{12}$$

If $\beta$ is acute, then we can see that using the Pythagorean theorem, we get $\|\mathbf{T} + \mathbf{u}\|_2 = \sqrt{(T + u\cos(\beta))^2 + (u\sin(\beta))^2}$. One can show that this also holds for obtuse $\beta$, in which case $u\cos(\beta)$ is negative. Substituting this in the equation above, we get our Eq. 7 as

$$\cos(\alpha) = \frac{T\cos(\beta) + u}{\sqrt{(T + u\cos(\beta))^2 + (u\sin(\beta))^2}}. \tag{13}$$

### A.7.4 Alignment vs Magnitude in GALA

In this section, we will try to expand on the discussion of Sec. 5 to understand better how the proposed cosine distance criterion depends on the *magnitude* and the *alignment* of the current sample's update with respect to the total displacement made so far.

From Eq. 7, it is clear that computing the proposed cosine distance criterion for a given layer only involves $T$, $u$, and the angle $\beta$. This means that even though the layers may have a considerable number of parameters, we can always draw a diagram like the ones in Fig. 8 to represent the situation and compare the updates for performing layer selection.

To better understand the interaction between magnitude and alignment towards cosine metric, we consider an example where we have two layers, layer 1 and layer 2, and their corresponding total displacement has the same norm $T$. We consider an update $\mathbf{u}_1$ for layer 1 and an update $\mathbf{u}_2$ for layer 2 with: $u_1 < u_2$. While the update for layer 2 has a larger magnitude, the update $\mathbf{u}_1$ of layer 1 is more aligned with its displacement ($\beta_1 < \beta_2$). We observe that two scenarios can arise depending on the magnitude of the total displacement:

- **Scenario 1 of large** $T$: In this scenario, Cosine distance significantly depends on the *alignment* of the current sample and is less impacted by its *magnitude*. This scenario is more likely for most of

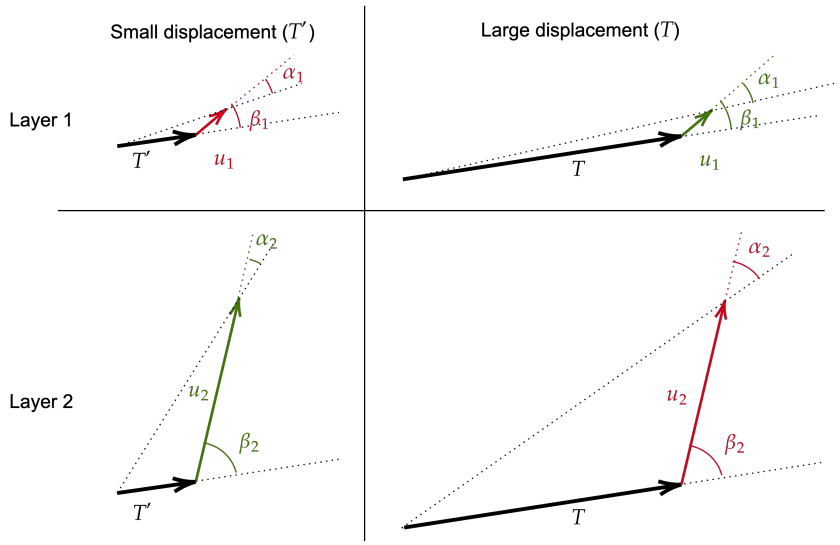



Figure 8: Effect of magnitude of $\mathbf{u}$ on cosine distance criterion. Consider two vectors such that $u_1$ is smaller than $u_2$ but is better aligned with its displacement. The rows illustrate the layers, and the columns denote the two scenarios. The vectors are shown in green if the cosine distance selects the layer, or else shown in red. **Left**: In scenario 2 of small displacements ($T'$), the update's magnitude can dominate the criterion, and GALA selects $u_1$. **Right** In scenario 1 of large displacements ($T$), alignment becomes crucial, and GALA selects $u_2$.

the samples during TTA. This can also be viewed similarly to performing an exploit strategy in a given update direction, i.e., after having seen a certain number of samples, we allow adaptation of a layer if the gradients for new updates are well aligned with the previously seen updates.

- **Scenario 2 of small** $T$: In this scenario, Cosine distance significantly depends on the *magnitude* of the current sample and is less impacted by its *alignment* with the total displacement. This can occur for the very first samples in a few cases or if the gradients on a layer for a particular sample are much larger than usual. Similarly, we can view it as performing a form of pure exploration strategy over the update directions.

Fig. 4 (left) and Fig. 8 visualize the two scenarios for the above example.

## A.8 Discussion

In this paper, we introduce Gradient Aligned Layer Adaptation (GALA), a novel layer selection framework explicitly designed for Test Time Adaptation (TTA). Our comprehensive study reveals that layers in neural networks exhibit varying receptiveness to adaptation, and the optimal set of layers for adaptation depends on both the specific distribution shift and the loss function employed during inference. Building on these insights, we propose GALA, a dynamic layer selection criterion that ranks layers based on gradient alignment, effectively mitigating overfitting and performance degradation. Extensive experiments across diverse datasets, model architectures, and TTA losses demonstrate GALA's superior performance compared to existing methods, including standard *ERM*, *all-layers* adaptation, and other layer selection baselines.

The simplicity and versatility of GALA enable seamless integration with existing TTA loss functions, making it a valuable tool for enhancing the adaptability and reliability of deep learning models in real-world applications. Beyond its immediate impact on TTA, our work opens up new avenues for future research in areas where regularization is crucial for learning stability, selective parameter updates could be beneficial, or gradient-aligned feature learning might offer additional advantages. GALA not only advances the field of TTA but also contributes to the broader goal of developing more robust and adaptive AI systems.