The Golden Subspace: Where Efficiency Meets Generalization in Continual Test-Time Adaptation

Supplementary Material

The supplementary material is organized as follows.

- Appendix A introduces the notation used throughout the paper.
- Appendix B provides the full theoretical proof of the existence of the golden subspace.
- Appendix C presents additional experimental details and results, including the detailed experimental setup, analyses under different batch sizes and hyperparameter settings, comprehensive efficiency analysis, additional qualitative segmentation results, and long-term generalization experiments.

A. Notations

To facilitate reading, Table S1 provides the mathematical symbols that appear throughout the text and their meanings.

Table S1. Notation Summary.

Symbol	Meaning
D_S	Labeled source-domain dataset
D_T	Unlabeled target-domain data stream
X_t	Mini-batch of target samples at step t
g_θ	Full model composed of feature extractor and classifier
g_ϕ	Pretrained feature extractor (frozen backbone)
h_ψ	Classifier head producing logits
F_t	Feature matrix of batch t
f	Feature vector of a single sample
W	Linear classifier weight matrix
P_c	Prototype of class c in the source domain
P	Matrix of all class prototypes
V_t	Basis of the adaptive subspace (top-r eigenvectors)
S_t	Learnable scaling vector in the subspace
u	Coordinates of feature f in the subspace
$\tilde{u}$	Scaled coordinates after subspace adaptation
$A(f)$	Adapter transformation applied to feature f
F_t^{adapt}	Adapted feature matrix for batch t
$\hat{G}_t^{(b)}$	Mini-batch estimate of the AGOP matrix
g_i	Gradient proxy with respect to feature f_i
M_t	Set of high-confidence samples in batch t
G_t	Exponential moving average (EMA) of AGOP
Q, Λ	Eigenvectors and eigenvalues of G_t
$\kappa(k)$	Cumulative spectral energy ratio
Y_t	Student model logits
Y_t^{ema}	EMA teacher logits
Y_t^+	Logits from augmented views
$\mathcal{L}_{st}$	EMA consistency loss
$\mathcal{L}_{cont}$	Prototype-based contrastive loss
$\mathcal{L}$	Total training objective
θ^{ema}	Parameters of the EMA teacher model
α, τ, r	EMA decay, confidence threshold, subspace rank
m	Momentum coefficient for EMA updates

B. Detailed Analysis of GOLD

B.1. Proof: Existence of the golden subspace

We begin by formalizing the claim that the minimal feature perturbation required to produce a prescribed logit change must lie in the row-space of the linear classifier. This statement underlies the notion of a "golden subspace" for test-time feature adaptation.

Assumption B.1 (Linear classifier)**.** *The classification head is linear in features: for any feature vector $f \in \mathbb{R}^L$ the pre-softmax logit vector satisfies*

$$z = Wf,$$

where $W \in \mathbb{R}^{C\times L}$ is the classifier weight matrix (the last fully-connected layer).

Proposition B.1 (Minimal-norm feature change)**.** *Given a desired logit change $\Delta y \in \mathbb{R}^C$, the constrained minimization*

$$\min_{\Delta f\in\mathbb{R}^L} \ \tfrac{1}{2}\|\Delta f\|_2^2 \quad \textit{s.t.} \quad W\Delta f = \Delta y$$

has the unique minimal-norm solution

$$\Delta f^\star = W^+\Delta y,$$

where W^+ denotes the Moore–Penrose pseudoinverse of W. In particular, $\Delta f^\star \in \mathrm{row}(W)$, i.e. the column space of $W^\top$.

Proof. Form the Lagrangian

$$\mathcal{L}(\Delta f, \lambda) = \tfrac{1}{2}\|\Delta f\|_2^2 + \lambda^\top(W\Delta f - \Delta y), \qquad \lambda \in \mathbb{R}^C.$$

Stationarity in Δf yields $\Delta f + W^\top\lambda = 0$, hence $\Delta f = -W^\top\lambda$. Inserting this into the constraint gives $-(WW^\top)\lambda = \Delta y$. Solving (via the pseudoinverse on the potentially rank-deficient matrix $WW^\top$) yields

$$\lambda = -(WW^\top)^+\Delta y,$$

and therefore

$$\Delta f^\star = -W^\top(WW^\top)^+\Delta y = W^+\Delta y,$$

using the identity $W^+ = W^\top(WW^\top)^+$. Equivalently, if $W = U\Sigma V^\top$ is a compact SVD, then

$$\Delta f^\star = V\Sigma^+U^\top\Delta y,$$

which has no component in $\ker(W)$ and is the unique minimal-norm feasible vector. Uniqueness follows because any feasible Δf can be decomposed as $\Delta f = \Delta f^\star + v$ with $v \in \ker(W)$, and $\|\Delta f\|_2^2 = \|\Delta f^\star\|_2^2 + \|v\|_2^2 \geq \|\Delta f^\star\|_2^2$. □

Batch extension. The same argument applies column-wise: for a batch of B samples let $\Delta Y \in \mathbb{R}^{C\times B}$ and $\Delta F \in \mathbb{R}^{L\times B}$. The Frobenius-norm constrained problem

$$\min_{\Delta F} \tfrac{1}{2}\|\Delta F\|_F^2 \quad \text{s.t.} \quad W\Delta F = \Delta Y$$

has the solution $\Delta F^\star = W^+\Delta Y$. Thus the minimal collective feature change again lies in the row-space of W.

The implication is direct: to realize logit (or prediction) corrections while minimally perturbing features, it suffices to search for updates inside the subspace spanned by the rows of W (equivalently the columns of $W^\top$). We refer to this subspace as the *golden subspace*.

B.2. Adapter Design and Pseudo-Label Robustness

Why an adapter? We adopt an adapter-style design to meet the core constraint of CTTA: *online adaptation must be both efficient and stable under non-stationary test streams*. Updating the full backbone (or a large fraction of parameters) can improve short-term fitting but often harms efficiency and exacerbates drift/forgetting as the stream evolves. In contrast, a lightweight adapter provides a structured and low-dimensional update interface: it preserves most pretrained representations while allowing the model to quickly adjust along a compact set of directions that are most relevant to the current target shift. This design choice is aligned with continual learning insights that emphasize *controlling the effective update subspace* to improve robustness to stream order and to reduce misleading gains that arise from unstable adaptation dynamics [6, 7, 22, 56]. In GOLD, the adapter is further constrained to operate in a low-rank *golden subspace*, so that adaptation is concentrated on a minimal set of feature directions rather than dispersed across the entire representation space.

Robustness of confidence-based pseudo labels for AGOP. AGOP relies on pseudo labels derived from the model's predictions to form a sample-wise gradient outer product estimate. A natural concern is that noisy pseudo labels may corrupt the estimated directions and lead to drift. We justify the robustness of our design from two complementary aspects.

(i) Built-in safeguards. GOLD includes three mechanisms that jointly stabilize pseudo-label-based estimation. *(a) Warm-start initialization.* We initialize the subspace estimate with the theoretically motivated prior $G_0 = W^\top W$ (Sec. 3.2), which anchors early updates to the pretrained classifier geometry when the model is least adapted and pseudo labels are potentially noisier. *(b) EMA smoothing with self-training consistency.* We use an exponential moving average (EMA) teacher to generate more stable predictions and enforce self-training consistency (Eq. 17), which reduces the variance of pseudo labels across adjacent samples/batches and discourages abrupt direction changes. *(c) Low-rank/subspace constraint.* By restricting updates to a compact low-rank subspace, the estimator is regularized implicitly: even if individual pseudo labels are imperfect, their influence is projected onto a small set of directions, mitigating error accumulation and reducing forgetting/drift over long streams.

(ii) Empirical sensitivity. We further validate robustness by sweeping the confidence threshold used to filter pseudo labels. As shown in Fig. S1, GOLD remains insensitive across a wide range of thresholds: even with a relatively low threshold, the performance drop is only $\sim$0.2. This suggests that the proposed safeguards effectively prevent noisy pseudo labels from dominating the AGOP-based subspace maintenance, leading to stable adaptation in practice.

C. Supplementary Experiments

This section supplements the main paper with additional empirical analyses on (i) the sensitivity of GOLD to **test-time batch size**, (ii) **hyper-parameter robustness**, (iii) **comprehensive efficiency** (trainable parameter ratio, FLOPs, and peak GPU memory), and (iv) additional **segmentation** results and qualitative comparisons. Unless otherwise specified, all experiments use the same pretrained backbones and CTTA evaluation protocol as in the main text. For corruption benchmarks, we report mean performance over the full benchmark and follow the same metrics and legend definitions as in the main paper.

C.1. Detailed Experimental Setup

Datasets. We evaluate GOLD on three standard continual test-time adaptation benchmarks: CIFAR-10-C, CIFAR-100-C, and ImageNet-C. CIFAR-10-C and CIFAR-100-C are corrupted versions of the original CIFAR-10 and CIFAR-100 test sets, respectively, while ImageNet-C is constructed from the ImageNet validation set with a diverse set of synthetic corruptions. Following the standard protocol, we consider 15 corruption types at severity level 5, including *gaussian_noise*, *shot_noise*, *impulse_noise*, *defocus_blur*, *glass_blur*, *motion_blur*, *zoom_blur*, *snow*, *frost*, *fog*, *brightness*, *contrast*, *elastic_transform*, *pixelate*, and *jpeg_compression*. We adopt the one-pass online adaptation setting, where each target sample is observed only once in a streaming manner without revisiting previous target data.

Parameter settings. For a fair comparison, all methods use the same optimization setup during test-time adaptation.

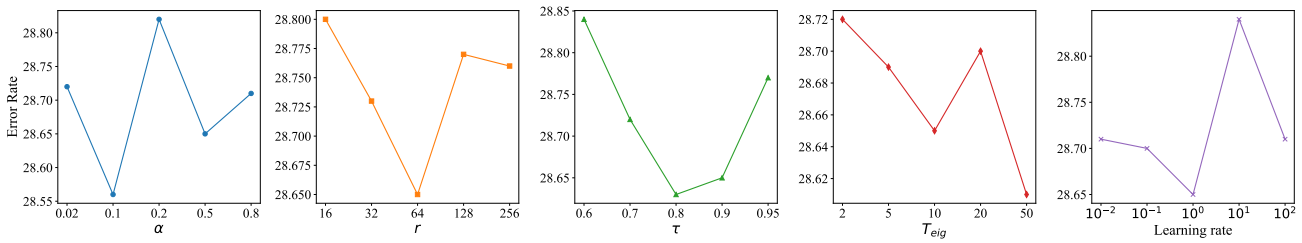

Figure S1. Hyper-parameter robustness: representative results from the sweep described in Section C.3. Each subplot shows the evaluated metric as a function of a single hyper-parameter while other hyper-parameters are held near their default values.

Figure S2. Detailed segmentation comparisons (cropped views). White boxes indicate regions shown at higher resolution to emphasize the qualitative gains of GOLD in challenging conditions (shadows, reflections, fog/night).

We update the affine parameters of normalization layers, including BatchNorm, LayerNorm, and GroupNorm. Concretely, for each normalization layer, we optimize only its `weight` and `bias` parameters. For all methods, we use Adam as the optimizer with learning rate 1×10^{-3}, weight decay 0, and one adaptation step per batch. We use Adam with `betas` $= (0.9, 0.999)$.

Augmentation details. For test-time augmentation, we follow a unified transformation pipeline consisting of Gaussian blur, center crop, random horizontal flip, additive Gaussian noise, and clipping to the valid input range. Specifically, we use `GaussianBlur` with kernel size 5 and sigma sampled from $[0.001, 0.25]$ for the soft setting and from $[0.001, 0.5]$ otherwise, followed by `CenterCrop`, `RandomHorizontalFlip` with probability 0.5, additive Gaussian noise, and a final clipping operation to $[0, 1]$.

Source models. Following CoTTA, we employ standard pre-trained source models for different benchmarks: WideResNet-28 [52] for CIFAR10-C, ResNeXt-29 [51] for CIFAR100-C, and ResNet-50 [13] for ImageNet-C. These source models are fixed before test-time adaptation and serve as the initialization for all compared methods.

Source prototype extraction. We extract source prototypes offline from the source training set before online adaptation. Specifically, we first feed all source samples through the feature extractor and collect their intermediate feature representations together with the corresponding class labels. For feature tensors with spatial dimensions, we flatten them into one-dimensional vectors. Then, for each class, we compute the class prototype as the mean feature vector of all source samples belonging to that class:

$$\mathbf{p}_c = \frac{1}{N_c} \sum_{i:y_i=c} \mathbf{f}_i,$$

where $\mathbf{f}_i$ denotes the extracted source feature, y_i is its label, and N_c is the number of source samples in class c. If a class has no source samples, we assign it a zero prototype vector. All source prototypes are computed once before adaptation and kept fixed throughout the online test-time adaptation process.

C.2. Effect of batch size

In the main text we reported results obtained with a batch size of 200 on CIFAR-C and 64 on ImageNet-C. Here we study the influence of test-time batch size on performance by evaluating GOLD and several baseline methods at batch sizes $\{4, 16, 32, 64\}$. Table S2 summarizes the results.

Discussion. The table shows that GOLD is stable across a wide range of batch sizes. On CIFAR-10-C, GOLD attains the best metric among compared methods for all evaluated batch sizes (columns 4–64). On CIFAR-100-C

Table S2. Performance under different test-time batch sizes. Results are shown for CIFAR-10-C, CIFAR-100-C and ImageNet-C. Columns report the evaluated batch size. The best value in each column (among the listed methods) is typeset in **bold**.

	CIFAR10-C				CIFAR100-C				ImageNet-C			
Method	4	16	32	64	4	16	32	64	4	16	32	64
TENT [42]	86.9	61.9	40.2	22.9	98.3	96.2	92.9	86.1	99.0	82.5	64.5	62.6
Ada [9]	34.5	19.7	17.7	17.4	90.1	54.6	41.3	35.4	98.4	81.9	69.9	65.5
CoTTA [44]	52.6	26.9	21.3	17.5	68.8	39.1	36.7	34.3	97.5	82.0	64.8	62.7
RMT [13]	51.4	26.5	20.0	16.7	87.4	48.7	39.1	32.7	99.4	83.4	68.7	60.2
SANTA [8]	39.5	19.5	17.5	16.8	58.5	35.8	33.3	32.5	98.7	68.0	62.8	60.1
GOLD (ours)	**31.7**	**18.9**	**17.0**	**16.4**	**57.8**	**35.1**	**32.8**	**31.9**	**97.3**	69.5	**62.1**	**59.3**

and ImageNet-C GOLD is competitive with state-of-the-art baselines; in particular GOLD achieves very strong performance at medium-to-large batch sizes on ImageNet-C (see bold entries). These results indicate that the proposed AGOP-based subspace estimation and the low-rank adaptation scheme are effective even when the number of high-confidence samples per batch is relatively small. In practice we recommend using batch sizes of at least 16–32 when computational resources allow, though reasonable performance is still obtained with batch size 4.

C.3. Hyper-parameter robustness

We performed an extensive hyper-parameter sweep to assess stability. The swept ranges were:

- EMA momentum $\alpha \in \{0.02, 0.1, 0.2, 0.5, 0.8\}$;
- retained rank $r \in \{16, 32, 64, 128, 256\}$;
- confidence threshold $\tau \in \{0.6, 0.7, 0.8, 0.9, 0.95\}$;
- eigendecomposition period $T_{\text{eig}} \in \{2, 5, 10, 20, 50\}$ (in batches);
- learning rate for the small adaptive parameter set (scaling vector s and optional BN adapters) $\eta \in \{10^{-2}, 10^{-1}, 1, 10, 10^{2}\}$.

Figure S1 visualizes representative slices of this sweep (one plot per dataset / metric). The main conclusions are:

1. **Overall robustness.** GOLD exhibits small performance variation over wide ranges of α and η; extreme values (very large η or α) can destabilize adaptation but are easy to detect.
2. **Rank trade-off.** Increasing r generally improves performance up to a point; in our experiments r between 32 and 128 provides a good balance between accuracy and computational cost.
3. **Confidence threshold.** A moderate threshold τ (e.g. 0.8) mitigates pseudo-label bias while retaining enough samples for stable AGOP estimation. Very high thresholds (e.g. 0.95) reduce the effective sample size and may increase variance.
4. **Eig update period.** Frequent eigen updates (small T_{eig}) track fast distribution shifts but increase compute; values in the range $T_{\text{eig}} = 5-20$ offered a good practical tradeoff in our workloads.

These observations guided the default hyper-parameter choices used throughout the main experiments and indicate that GOLD can be tuned with modest effort for new datasets or deployment constraints.

C.4. Comprehensive efficiency analysis.

We provide a systematic efficiency comparison on CIFAR100-C (Table S3 and Fig. 4 in the main paper) to quantify the practical cost of online CTTA. All methods are evaluated under the same backbone and input resolution, and we report three complementary metrics: **(i) trainable parameter ratio**, **(ii) FLOPs**, and **(iii) peak GPU memory**. Together, these metrics capture *what is updated* (parameter footprint), *how much computation is incurred per batch* (throughput cost), and *how much hardware budget is required* (deployment feasibility).

Trainable parameter ratio. This measures the fraction of parameters receiving gradients during adaptation. Full fine-tuning style CTTA methods (CoTTA, RMT, GTTA) effectively update the whole model, resulting in a 100% trainable ratio. In contrast, **SANTA** and **GOLD** operate with lightweight modules: only a small adapter and/or scaling parameters are optimized, leading to orders-of-magnitude fewer trainable parameters (0.365% and 0.373%, respectively). This is particularly important for online deployment where frequent gradient updates must be performed with minimal overhead and minimal risk of representation drift.

FLOPs. We report the per-batch floating-point operations to reflect end-to-end adaptation cost, including both forward and backward passes. **CoTTA** is the most expensive due to full-model updates and multi-view/self-ensemble style operations, reaching 7842.80 G FLOPs. **RMT** and **GTTA** reduce computation compared to CoTTA but still remain substantially heavier than adapter-based methods (1886.02 G and 2614.27 G). In contrast, **SANTA** and **GOLD** are significantly more efficient (1348.09 G and 1425.14 G), show-

ing that restricting updates to a compact subspace can deliver strong performance without the heavy compute of full-parameter adaptation. Notably, GOLD incurs only a modest additional FLOPs over SANTA due to maintaining and applying the golden-subspace projection.

Peak GPU memory. Peak memory reflects the maximum activation/gradient/storage footprint during online adaptation. Memory is a key bottleneck for real-time CTTA, especially when running larger backbones or higher resolutions. Full-update methods require storing gradients for most layers, leading to high memory usage (CoTTA: 10.88 GB; RMT/GTTA: 5.07–5.21 GB). Adapter-based methods are notably lighter (SANTA: 4.61 GB), and while **GOLD** uses slightly more memory (5.37 GB) due to additional subspace-related buffers, it remains far below full-model adaptation and fits comfortably within typical single-GPU deployment constraints.

Takeaway. Table S3 confirms that **GOLD achieves a favorable accuracy–efficiency trade-off**: it keeps the trainable parameter ratio below 0.4% while maintaining compute and memory footprints close to other lightweight methods, yet delivers stronger and more stable adaptation performance (main paper, Fig. 4). This supports our design principle that CTTA should prioritize *structured minimal updates*—adapting only along critical directions—to maximize both generalization and deployability.

Table S3. Comprehensive efficiency comparison.

Method	Trainable parameter ratio (%)	FLOPs (G)	Peak GPU memory (GB)
CoTTA	100	7842.80	10.88
RMT	100	1886.02	5.07
GTTA	100	2614.27	5.21
SANTA	0.365	1348.09	4.61
GOLD	0.373	1425.14	5.37

Runtime breakdown. To further clarify the practical overhead introduced by GOLD, we profiled the wall-clock time of its main components during online adaptation. On batches where eigendecomposition is performed, the forward and backward passes dominate the runtime, accounting for 45.7% and 50.8% of the total time, respectively. In contrast, the additional overhead introduced by AGOP computation is **only 0.3%**, while periodic eigendecomposition contributes 3.3%. These results indicate that the extra cost of GOLD mainly comes from the standard optimization steps shared by most adaptation methods, whereas the proposed subspace maintenance itself adds only a small overhead. This supports our claim that GOLD achieves strong adaptation performance with a favorable efficiency profile.

C.5. Expanded segmentation examples

We provide enlarged views of selected segmentation examples to highlight qualitative improvements obtained by GOLD. Figure S2 shows cropped regions (white bounding boxes) that emphasize the areas of largest visual difference between the pretrained baseline and the adapted model. Representative cases include:

- **Shadowed zebra crossings (Highway).** GOLD recovers occluded lane markings in regions where strong shadows cover parts of the road surface.
- **Puddles and water reflections (Rain).** The adapted model more reliably segments road surfaces and curbs in the presence of specular highlights and reflections.
- **Fog and low-visibility (Fog / Night).** GOLD improves detection of distant objects and preserves fine-grained foreground structure under heavy atmospheric degradation.

C.6. Long-term Generalization under Repeated Domain Exposure

Setup. To evaluate long-term generalization, we conduct an extended continual test-time adaptation experiment on CIFAR10-C. Instead of passing through the target stream only once, we repeatedly expose the model to the same corrupted target data for 10 rounds in sequence. This setting allows us to examine whether an adaptation method remains stable over prolonged online updates, or gradually suffers from error accumulation and representation drift. In particular, it reflects the method's ability to preserve robust generalization under repeated domain exposure, which is crucial for realistic deployment scenarios where incoming data may exhibit persistent or recurring shifts over a long time horizon.

Results. Table S4 reports the average error rates of different methods under the 10-round continual adaptation setting on CIFAR10-C. Compared with prior CTTA methods, GOLD achieves the best performance with an error rate of 14.15%, showing stronger stability and better long-term generalization under sustained adaptation. These results suggest that restricting updates to the dynamically estimated golden subspace can effectively mitigate parameter drift and maintain adaptation quality over long horizons.

Table S4. Long-term generalization comparison on CIFAR10-C under 10 rounds of continual adaptation. Lower is better.

Method	CoTTA	RMT	BeCoTTA	SANTA
Error (%)	16.21	16.73	16.28	16.29
Method	EATA	SAR	ViDA	**GOLD**
Error (%)	17.84	20.31	17.24	**14.15**