PROTOTTA: EXPLAINABLE TEST-TIME ADAPTATION VIA PROTOTYPE GUIDANCE

Anonymous authors
Paper under double-blind review

ABSTRACT

Deep networks that rely on prototypes—interpretable representations that can be related to the model input—have gained significant attention for balancing high accuracy with inherent interpretability, which makes them suitable for critical domains such as healthcare. However, their prototypes are limited by their reliance on training data, which hampers their robustness to distribution shifts. While test-time adaptation (TTA) improves the robustness of deep networks by updating parameters and statistics, the prototypes of interpretable models have not been explored for this purpose. We introduce ***ProtoTTA***, a general framework that minimizes the binary entropy of prototype similarities to encourage confident, prototype-specific activations, while adaptively balancing this objective with output entropy based on a label-free estimate of prototype reliability. We stabilize these updates by geometric filtering and weighting by prototype importance and model confidence. Experiments across four prototypical backbones on four benchmarks in fine-grained vision, histopathology, and NLP demonstrate that ProtoTTA improves robustness over standard output entropy minimization while restoring the correct grounding of prototypes. We also introduce novel interpretability metrics and a vision-language model (VLM) evaluation framework to quantify and explain our test-time updates, finding positive associations between prototype alignment and VLM-rated reasoning quality. Code will be released upon publication.

1 INTRODUCTION

Prototype-based deep networks are attractive for safety-critical applications by their balancing of high accuracy with inherent interpretability. Unlike standard black-box models, these architectures provide explainability of internal representations by classifying inputs through similarity matching with learned prototypes, enabling intuitive "this looks like that" reasoning with direct evidence for predictions (Chen et al., 2019). This transparency has driven growing adoption in high-stakes applications, particularly healthcare (Wei et al., 2024; Vaseli et al., 2023; Sethi et al., 2025), where model accuracy and understanding model decisions are both important.

Like other deep networks, prototype-based models are vulnerable to distribution shifts and their reliance on case-based comparisons could make them particularly susceptible. While Test-Time Adaptation (TTA) (Wang et al., 2020) and test-time training (Sun et al., 2020) mitigate shift using unlabeled target data, they treat models as black boxes, optimizing solely over output logits or global feature statistics. However, they are blind to the failure mode unique to these prototypical architectures: under shift, the prototype selection mechanism breaks down by activating irrelevant prototypes and suppressing correct ones. Existing TTA methods overlook the intermediate prototype signals that distinguish these models from conventional deep networks.

This gap raises three questions in our work. **(RQ1)** *Can a model's inherent interpretability signals drive more reliable test-time adaptation than output-level signals alone?* **(RQ2)** *Beyond accuracy, how do different TTA methods reshape the model's internal reasoning, and can this be measured?* **(RQ3)** *Can these reasoning changes be explained automatically, in natural language, so that adaptation itself becomes more interpretable?*

To answer RQ1, we introduce *ProtoTTA*, a TTA framework designed for prototype-based architectures. It minimizes the binary entropy of prototype-similarity activations to encourage decisive

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

matches on shifted data. An adaptive factor λ interpolates this objective with output entropy: reliable prototype activations receive greater weight, while weakly discriminative ones favor output supervision. To ensure stability, adaptation is bounded by geometric filtering on reliable samples, regularized by prototype importance and prediction confidence. The same prototype signals also let us observe adaptation by observing their explicit reasoning trace. For RQ2, we introduce interpretability-aware metrics that assess consistency, prototype alignment, and prediction stability to compare how TTA methods change the model's reasoning. For RQ3, we introduce a vision-language model (VLM)-based evaluation framework that scores and explains these changes through language and prototype evidence (Figures 2 and 5) to help interpret the updates.

Contributions. (i) We present *ProtoTTA*, a test-time adaptation framework that adaptively combines prototype-level and output-level entropy. **(ii)** *ProtoTTA* consistently outperforms TTA baselines across four prototypical backbones and four benchmarks spanning fine-grained vision, histopathology, and NLP, while remaining fully test-time. **(iii)** We introduce interpretability-aware metrics and show that ProtoTTA generally improves clean–corrupted consistency and correct-class prototype alignment. **(iv)** We propose a VLM-based evaluation framework for explainable TTA and find positive associations between prototype alignment and VLM-rated reasoning quality.

2 Related Work

Prototype-Based Interpretable Models. The foundational model, ProtoPNet (Chen et al., 2019), introduced the concept of classifying images by comparing input feature patches to learned prototypes. However, ProtoPNet relies on spatially rigid prototypes, limiting its ability to capture geometric variations. To address this, Deformable ProtoPNet (Donnelly et al., 2022) replaced single rigid prototypes with flexible sub-prototypes that adapt their positions to better match input features. Subsequently, ProtoViT (Ma et al., 2024) extended this paradigm to Vision Transformer (ViT) (Dosovitskiy et al., 2021) backbones with coherence-aligned sub-prototypes for capturing complex geometric variations, and ProtoPFormer (Xue et al., 2024) adapted part-prototype reasoning to the global and local token structure of ViTs. The paradigm has also moved beyond vision: ProtoLens (Wei & Zhu, 2025) brings prototype-based reasoning to text classification at the sub-sentence level. The field continues to evolve with innovations such as Hyperbolic Hierarchical Part Prototypes (Vaseli et al., 2025), which demonstrates sustained community interest in advancing prototype-based architectures. Despite this architectural progress, these models are not explicitly designed to handle distribution shift, and their robustness in this setting remains largely unexplored.

Test-Time Adaptation. TTA (Liang et al., 2025) has emerged as a paradigm to dynamically adjust models at inference using unlabeled target-domain data. Foundational methods such as Tent (Wang et al., 2020) update normalization parameters via entropy minimization, while subsequent approaches have improved stability and efficiency through active sample selection (EATA) (Niu et al., 2022), sharpness-aware minimization (SAR) (Niu et al., 2023), or enforcing consistency across augmented views (MEMO) (Zhang et al., 2022). Lately, ReservoirTTA (Vray et al., 2025) proposed maintaining domain-specialized model reservoirs with online style clustering for recurring distribution shifts. Beyond standard discriminative tasks, TTA has expanded to generative models (Prabhudesai et al., 2023) and even LLM agents that adapt to unseen environments by adjusting to local observation formats (Chen et al., 2025). These methods primarily adapt through outputs or global feature statistics, rather than explicitly restoring the prototype evidence underlying a prediction.

Interpretability-Aware Adaptation. The intersection of TTA and model interpretability remains largely unexplored. Recent work such as TIDE (Agarwal et al., 2025) shows that training with local concept supervision can enable test-time correction, but it requires specialized training pipelines and externally generated concept annotations. In the context of prototype-based models, current approaches overlook the interpretable intermediate signals these architectures already expose. Whether such signals can drive adaptation itself, how adaptation reshapes a model's reasoning, and whether these reasoning changes can be evaluated and explained automatically has not been studied. To address this gap, we propose *ProtoTTA* that operates directly on prototype activations, requires no specialized training or external annotations. We assess changes in prototype-based reasoning using interpretability-aware metrics and a VLM framework that narrates adaptation dynamics and evaluates agreement between these metrics and VLM-rated reasoning quality.

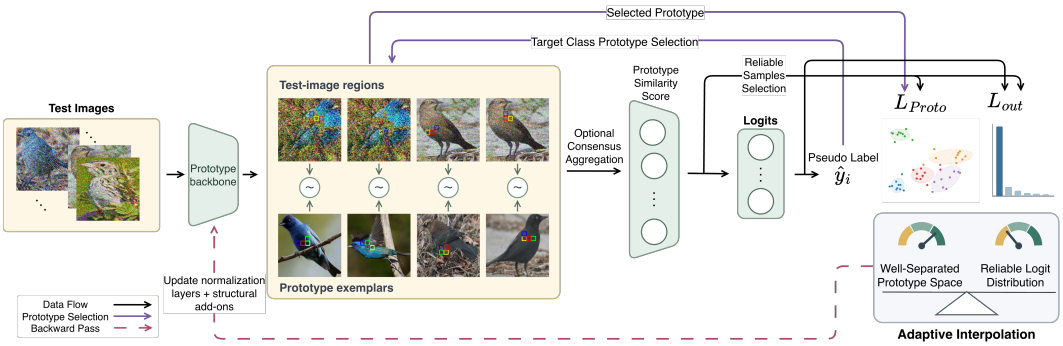


Figure 1: Overview of the *ProtoTTA* pipeline. For reliable test samples, prototype similarities and output logits define complementary entropy losses. Prototype-space separation and output confidence determine their adaptive interpolation, which guides test-time updates.

3 Method: Prototype and Output Updates

Prototype-based architectures provide three forms of granular interpretability absent in black-box models: (i) similarity scores quantifying input-prototype matching (prototype activations), (ii) prototype-to-class weights in the final classification head, and (iii) spatial localization binding prototypes to specific input regions. Figure 1 illustrates how *ProtoTTA* uses these signals to recalibrate prototype matching under shift. We encourage decisive prototype activations and adaptively balance this objective with output entropy.

3.1 Prototype-Level Entropy Minimization

Distribution shifts can corrupt the prototype selection mechanism: the model activates irrelevant prototypes and suppress correct ones, which degrades both accuracy and interpretability. Our adaptation strategy counters this by encouraging decisive, grounded prototype matching. To prevent unreliable samples from destabilizing adaptation, we update only on a reliable set $\mathcal{R}$ selected by a backbone-specific geometric statistic. For each test sample $x_i \in \mathcal{R}$, the forward pass yields a pseudo-label $\hat{y}_i$, and we focus our adaptation specifically on the set of target prototypes $\mathcal{A}_i := \mathcal{P}_{\hat{y}_i}$ associated with this pseudo-label. Exact mappings and thresholds are given in Appendix L.

Prototype activations are similarity scores rather than probabilities, and each prototype can legitimately exhibit strong or weak similarity. We therefore map raw scores s_{ip} through a backbone-specific transformation ϕ_p, apply a small numerical clamp, and minimize the binary entropy of each mapped activation independently. This encourages decisive similarities near 0 or 1 rather than ambiguous mid-range values. In this mapped space, a value of 0.5 corresponds to maximum entropy (equivalent to cosine similarity of 0 for the symmetric cosine mappings). The per-sample prototype loss and its batch average are:

$$\ell_i^{\text{proto}} = \sum_{p \in \mathcal{A}_i} w_{ip} H_{\text{b}}(\bar{s}_{ip}), \qquad \mathcal{L}_{\text{proto}} = \frac{1}{|\mathcal{R}|} \sum_{i \in \mathcal{R}} c_i \ell_i^{\text{proto}}. \tag{1}$$

where $H_{\text{b}}(\bar{s}_{ip}) = -\bar{s}_{ip}\log(\bar{s}_{ip}) - (1-\bar{s}_{ip})\log(1-\bar{s}_{ip})$ is the binary entropy of the mapped prototype activation, c_i is the confidence score of sample x_i, and w_{ip} represents the prototype weight from the final classification head for the pseudo-label $\hat{y}_i$. Minimizing this objective suppresses low-activation noise while reinforcing strong associations, even within a class: for each prototype, the sign of the gradient of H_{b} changes when its activation crosses 0.5 (Details in Section 3.4). For sub-prototypes, maximum pooling is sensitive to outliers while averaging weakens decisive signals by mixing them with low-activation noise; we instead aggregate the k strongest scores using Top-K Mean. Shared-prototype architectures lack hard class membership, so we derive importance weights and soft targets from the classifier head and use importance-weighted binary cross-entropy (Appendix H).

3.2 Adaptive Interpolation Between Prototype and Output Entropy

Prototype entropy provides a finer-grained adaptation signal than output entropy because it can adjust individual prototype activations within the same class, but its reliability varies across architectures and test batches. A well-separated prototype space provides reliable intermediate supervision; poor calibration can make even seemingly confident activations produce unstable, non-discriminative gradients. Output entropy provides a coarser, complementary signal that is less sensitive to such prototype-level noise. Our analysis in Section 3.4 motivates combining them according to their reliability rather than fixing the balance for each backbone.

Let ℓ_i^{proto} and $\ell_i^{\text{out}} = H(\boldsymbol{p}_i)$ denote the per-sample prototype and output entropy losses, respectively. Both losses are evaluated over the reliable set $\mathcal{R}$ selected by geometric filtering and use the same confidence weight c_i. We remove the substantial scale disparity between these two signal gradients by computing the batch gradient norm of each component:

$$g_j = \left\| \nabla_\theta \frac{1}{|\mathcal{R}|} \sum_{i \in \mathcal{R}} c_i \ell_i^j \right\|_2, \qquad j \in \{\text{proto}, \text{out}\}, \tag{2}$$

where θ denotes the parameters adapted at test time. The resulting norms are treated as fixed scaling factors during the final optimization step. We then define the unified objective as:

$$\mathcal{L}_\lambda = \frac{1}{|\mathcal{R}|} \sum_{i \in \mathcal{R}} c_i \left[\lambda_i \frac{\ell_i^{\text{proto}}}{g_{\text{proto}} + \epsilon} + (1 - \lambda_i) \frac{\ell_i^{\text{out}}}{g_{\text{out}} + \epsilon} \right], \tag{3}$$

where $\epsilon > 0$ ensures numerical stability. We recompute c_i, λ_i, and g_j from the current batch and treat them as fixed coefficients during differentiation. When $\lambda_i = 0$, adaptation relies entirely on output entropy; when $\lambda_i = 1$, it relies entirely on prototype entropy; intermediate values interpolate between the two signals.

Label-free estimation of the interpolation coefficient. Without target labels or source data, we estimate λ_i from two complementary measures of prototype reliability in the current test batch. The first estimate, the *activation-margin coefficient*, measures the decisiveness of the predicted-class prototype activations. Let $\mathcal{T}_i$ contain the top-k normalized prototype activations associated with the pseudo-label. We define:

$$\lambda_i^{\text{act}} = \text{clip}\left(\frac{1}{k\delta_0} \sum_{p \in \mathcal{T}_i} \left| \bar{s}_{ip} - \frac{1}{2} \right|, \, 0, \, 1 \right) \tag{4}$$

Here, k is the number of strongest predicted-class prototype activations considered, and $\delta_0 > 0$ is a reference margin that determines when the coefficient reaches its maximum value. Activations near $\frac{1}{2}$ can cross the prototype-gradient sign boundary under small perturbations and receive a small coefficient λ_i^{act}; larger margins indicate greater local directional stability. As shown in Section 3.4 and Remark 1, the activation margin is precisely the perturbation radius within which the coordinate-level gradient sign remains unchanged. It is used as a label-free proxy for directional stability rather than as an exact estimator of gradient variance. However, activation decisiveness alone can be misleading: uniformly large or saturated activations need not discriminate classes but be far from $\frac{1}{2}$. Our second estimate, the *relative-evidence coefficient*, therefore compares prototype and output class separation.

For each class, we aggregate the top-k activations of its associated prototypes. We then compute a normalized prototype margin r_i^{proto} between the prototype evidence for the predicted class and its strongest competing class. Similarly, r_i^{out} measures the normalized margin between the two largest output probabilities. Their relative contribution determines:

$$\lambda_i^{\text{rel}} = \frac{r_i^{\text{proto}}}{r_i^{\text{proto}} + r_i^{\text{out}} + \epsilon} \tag{5}$$

This coefficient is large when prototypes provide stronger class separation than the output distribution. Unlike λ_i^{act}, this estimate depends on relative class evidence rather than absolute activation magnitude. More details appear in Appendix L.

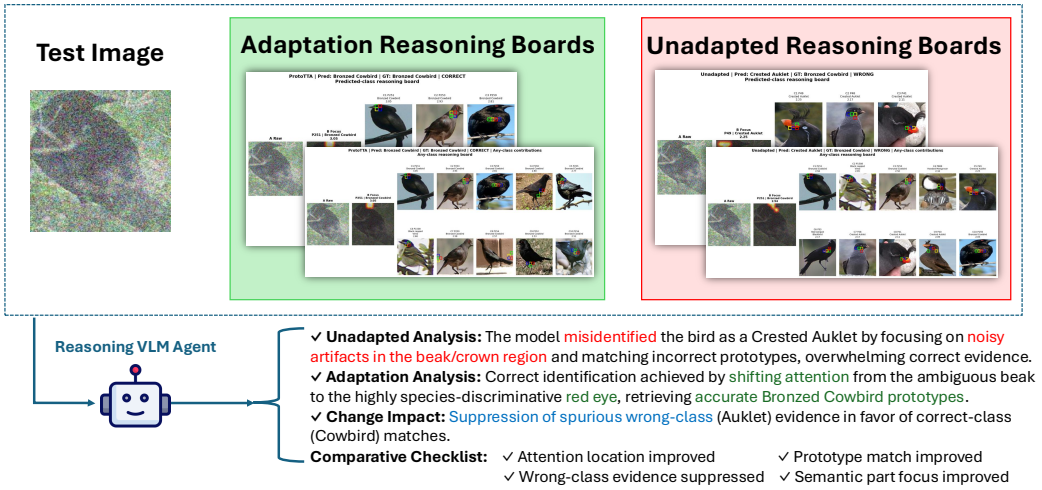


Figure 2: Explainable TTA via prototype reasoning boards. The unadapted model misclassifies the corrupted bird by focusing on noisy beak/crown artifacts and matching wrong-class prototypes. *ProtoTTA* shifts attention to the discriminative red eye, matches correct-class Bronzed Cowbird prototypes, and suppresses spurious evidence. A VLM automatically narrates this before/after change.

Test-time routing between the estimates. The two coefficients capture complementary notions of reliability. We split $\mathcal{R}$ into two interleaved subsets and measure the directional and magnitude agreement of each signal's gradients. A batch-level variable G selects λ_i^{act} when the prototype gradient is sufficiently stable relative to the output gradient, and selects λ_i^{rel} otherwise. A coverage guard also selects the activation-margin coefficient when saturated output predictions conflict with low geometric-filter coverage.

$$\lambda_i = G\lambda_i^{\text{act}} + (1-G)\lambda_i^{\text{rel}}, \qquad i \in \mathcal{R}. \tag{6}$$

Here, G is shared across the batch, while λ_i remains sample-specific. Exact consistency scores, routing conditions, thresholds, and the coverage guard are given in Appendix L.

3.3 What Parameters to Update

Finally, we update normalization parameters to correct global shifts and fine-tune structural add-ons, specifically attention biases (for Transformers) or 1×1 convolutions (for CNNs). These targeted updates recalibrate the model's focus under domain shift while preserving the structural knowledge learned during training; updating all parameters at the default learning rate causes collapse under small test batches, while normalization-only updates lack the capacity to re-focus attention (ablations in Section 5). The full procedure and pseudocode is given in Appendix G. Appendix L gives the exact backbone-specific definitions.

3.4 Why Prototype-Level Entropy, and When to Trust It

Prototype-level supervision can adjust evidence that output entropy cannot distinguish locally. For classifier-facing activations $\boldsymbol{s}_i$, logits $h(\boldsymbol{s}_i)$, and any differentiable output loss F_i, the chain rule gives $\nabla_{\boldsymbol{s}_i} F_i = \boldsymbol{J}_{h,i}^\top \nabla_{\boldsymbol{z}_i} F_i$. Thus, output losses are locally insensitive to directions in $\ker(\boldsymbol{J}_{h,i})$. In contrast, the prototype loss has coordinate derivatives

$$\frac{\partial \ell_i^{\text{proto}}}{\partial s_{ip}} = w_{ip}\phi_p'(s_{ip}) \log \frac{1-\bar{s}_{ip}}{\bar{s}_{ip}}, \qquad p \in \mathcal{A}_i, \tag{7}$$

for differentiable, strictly increasing mappings ϕ_p and positive weights. It can reinforce strong activations and suppress weak ones within the same class, including directions invisible to the output loss (Proposition 1). This is a local statement at the prototype interface; parameter updates also depend on the parameter-to-activation Jacobian.

The margin $|\bar{s}_{ip} - \frac{1}{2}|$ measures the perturbation required to cross the prototype-gradient sign boundary, motivating λ_i^{act}. It does not guarantee class discrimination, which motivates the complementary relative-evidence estimate. Proposition 2 further shows that the minimum-error mixture of normalized prototype and output gradients depends on their errors and covariance. These errors are unobservable in TTA: our margins and half-batch consistency scores are practical proxies. Full statements and proofs appear in Appendix D.

3.5 Measuring How Adaptation Changes Reasoning

Standard TTA evaluation reports accuracy under shift, which answers whether adaptation helps but not how. Since prototype models expose their reasoning, we can ask a complementary question: does adaptation restore the model's original reasoning process, or does it merely reshape output statistics? We introduce interpretability-aware metrics (RQ2) and a VLM-based framework that explains adaptation dynamics in language (RQ3), using paired clean and corrupted inputs for offline measurement.

Weighted Prototype Alignment (PCA-W). PCA-W measures how much of the strongest prototype evidence supports the ground-truth class. For sample i, let $\mathcal{T}_i$ contain the top-k activated prototypes and define $c_{i,p} = a_{i,p}|W_{y_i,p}|$, where $a_{i,p}$ is the prototype activation and $W_{y_i,p}$ is its classifier weight for the ground-truth class y_i (shared-prototype rule is given in Appendix E.2). Across N test samples, higher scores indicate that more of the top-prototype contribution is assigned to the correct class:

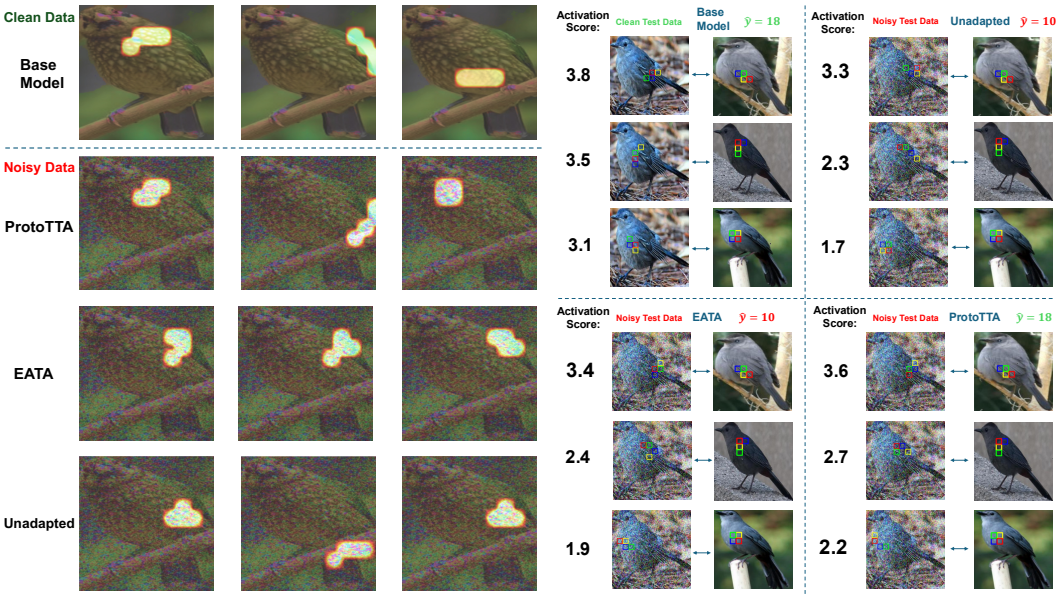
$$\text{PCA-W } (\%) = \frac{100}{N} \sum_{i=1}^{N} \frac{\sum_{p \in \mathcal{T}_i} c_{i,p}\, \mathbb{1}[\text{class}(p) = y_i]}{\sum_{p \in \mathcal{T}_i} c_{i,p}}. \tag{8}$$

Additional Metrics. We report *Prototype Activation Consistency (PAC)*, the cosine similarity between clean and adapted-corrupted prototype activations, and *Prediction Stability*, the prediction agreement between these paired inputs. Clean counterparts and ground-truth labels are used only for offline evaluation, not adaptation. Selection rate and relative speed characterize operational cost. Full definitions are provided in Appendix E.4.

VLM-Based Explanation Framework. Prototype-based TTA is inherently explainable: for every prediction we can assemble a *reasoning board* exposing the full evidence chain, which input regions are attended to, which training prototypes are retrieved, and how their contributions change under adaptation. For each prediction, we construct a *reasoning board* containing the corrupted input, predicted-class prototype matches with spatial focus, and the strongest any-class contributions. A VLM scores each board from 1–5 for *Focus Relevance*, *Prototype Match*, and *Overall Adaptation Quality* (Figure 5; protocol in Appendix E). The same agent can produce a comparative narrative between unadapted and adapted boards, describing exactly how adaptation altered the model's focus in language (Figure 2). The framework makes adaptation itself auditable and tests whether PCA-W is associated with an independent automated assessment of reasoning quality.

4 Experimental Setup

Backbones and Benchmarks. We evaluate four prototypical backbones. **(1)** On *CUB-200* (Wah et al., 2011) (200 classes), ProtoViT (Ma et al., 2024) uses DeiT-S/16 with 2,000 prototypes, each comprising four sub-prototypes (86.1% clean accuracy). **(2)** On *SICAPv2* (Silva-Rodríguez, 2020) prostate histopathology, ProtoPNet (Chen et al., 2019) uses VGG19-BN (Simonyan & Zisserman, 2014) and 50 prototypes (63.4% clean accuracy). **(3)** On *Stanford Dogs* (Khosla et al., 2011) (120 classes), ProtoPFormer (Xue et al., 2024) uses a dual-branch DeiT-S/16 with 1,800 prototypes (82.4% clean accuracy). Each vision benchmark uses the same 13 ImageNet-C corruptions at severity 5 (Hendrycks & Dietterich, 2019). **(4)** For NLP, ProtoLens (Wei & Zhu, 2025) uses *all-mpnet-base-v2* (Song et al., 2020) and 50 shared prototypes, trained on Yelp (94.0%) and tested on Amazon (91.9% clean accuracy). *Amazon-C* follows WildNLP (Rychalska et al., 2019) with five corruption types at four severities. Dataset details, similarity mappings, and corruption lists are in Appendix A.



(a) Spatial activation recovery **(b) Prototype-score recovery**

Figure 3: Recovery of clean prototype behavior under noise. **(a)** *ProtoTTA* restores spatially distributed activations across bird regions, closely resembling the model on clean data; but EATA and the unadapted model repeatedly concentrate on restricted region. **(b)** *ProtoTTA* shifts the top-3 prototype scores toward clean values and predicts correct class; both baselines predict the wrong class.

Table 1: Overall accuracy (%; mean ± SD across settings after three-seed averaging). * and ** denote Holm-adjusted $p < 0.05$ and $p < 0.01$ for each *ProtoTTA* variant against the strongest non-*ProtoTTA* baseline (two-sided paired t-test).

	Vision			NLP
Method	**ProtoViT** *CUB-200-C*	**ProtoPNet** *SICAPv2-C*	**ProtoPFormer** *Stanford Dogs-C*	**ProtoLens** *Amazon-C*
Unadapted	51.9 ± 13.6	31.4 ± 15.6	52.8 ± 11.0	80.81 ± 7.62
MEMO	52.4 ± 14.0	55.9 ± 4.6	54.1 ± 11.2	N/A
SAR	52.5 ± 13.3	55.8 ± 5.0	41.0 ± 11.5	77.98 ± 9.69
Tent	54.0 ± 13.3	54.3 ± 5.0	57.2 ± 9.1	79.74 ± 8.65
EATA	59.0 ± 11.3	55.7 ± 4.9	57.2 ± 8.9	80.79 ± 7.59
ProtoTTA (Fixed λ)	$\mathbf{60.3 \pm 11.0}^{**}$	$\mathbf{56.2 \pm 4.7}$	$\mathbf{57.5 \pm 8.8}$	$\mathbf{81.32 \pm 7.70}^{*}$
ProtoTTA (Adaptive λ)	$\underline{60.1 \pm 11.1}^{**}$	$\underline{56.2 \pm 5.0}$	$\underline{57.4 \pm 8.7}$	$\underline{81.31 \pm 7.79}^{**}$

Baselines and Optimization. We compare against Tent (Wang et al., 2020), EATA (Niu et al., 2022), SAR (Niu et al., 2023), and MEMO (Zhang et al., 2022), all applied to the same backbones. Optimization uses Adam (Kingma & Ba, 2017) and a single gradient update per batch in non-episodic settings; backbone-specific details are provided in Appendix L. Notably, ProtoTTA is entirely source-free, whereas EATA requires ≈2000 source samples to estimate its Fisher regularizer. We report two variants: the full *ProtoTTA* with the proposed adaptive λ_i, and *ProtoTTA-Fixed*, a diagnostic variant using the best fixed λ from the accuracy sweep, to show that the prototype-guided objective remains effective even without adaptive interpolation.

5 Results and Discussion

Effect of Adaptation Across Backbones. Both *ProtoTTA* variants outperform the strongest baseline on all four backbones (Table 1; per-corruption results in Appendix Tables 5 and 6). On ProtoViT, adaptive *ProtoTTA* gains 8.2 points over the unadapted model and 1.1 over EATA while remaining

Table 2: Interpretability and operational-cost analysis across all four backbones and datasets. Standard deviations are across corruption settings after seed averaging (three seeds). Relative speed is normalized to the unadapted run within each backbone; operational rankings exclude that reference. * and ** denote Holm-adjusted $p < 0.05$ and $p < 0.01$ for each *ProtoTTA* variant against the strongest non-*ProtoTTA* baseline (two-sided paired t-test).

Method	Interpretability Metrics			Operational Metrics	
	Semantic Consis. (PAC) ↑	Proto. Alignment (PCA-W) ↑	Prediction Stability ↑	Selection Rate ↓	Rel. Speed ↑
Backbone: ProtoViT (LN + Attention Biases) — Dataset: CUB-200-C					
Unadapted	88.6 ± 3.9	70.8 ± 11.2	$54.1 \pm 14.6\%$	0.0%	100.0%
MEMO	88.4 ± 4.5	70.3 ± 12.0	$54.7 \pm 15.1\%$	100.0%	4.1%
SAR	88.8 ± 3.8	71.6 ± 10.7	$54.9 \pm 14.3\%$	98.3%	48.4%
Tent	89.1 ± 4.1	73.8 ± 10.3	$56.7 \pm 14.5\%$	100.0%	94.3%
EATA	90.9 ± 3.0	77.7 ± 8.3	$61.9 \pm 12.3\%$	66.5%	**94.6%**
***ProtoTTA* (adaptive routing)**	$\mathbf{91.4 \pm 3.0}^{**}$	$\mathbf{79.2 \pm 8.2}^{**}$	$\mathbf{63.2 \pm 12.4\%}^{**}$	**54.6%**	68.9%
Backbone: ProtoPNet (BN + 1×1 Convs) — Dataset: SICAPv2-C					
Unadapted	89.8 ± 3.8	51.3 ± 5.9	$33.8 \pm 22.1\%$	0.0%	100.0%
MEMO	92.4 ± 1.4	61.4 ± 1.6	$58.0 \pm 7.0\%$	100.0%	45.5%
SAR	92.6 ± 1.5	61.8 ± 1.9	$58.3 \pm 7.8\%$	100.0%	22.6%
Tent	92.0 ± 1.6	61.2 ± 2.0	$54.9 \pm 7.2\%$	100.0%	45.4%
EATA	$\mathbf{92.6 \pm 1.5}$	61.8 ± 1.9	$58.2 \pm 7.7\%$	**3.0%**	**89.6%**
***ProtoTTA* (adaptive routing)**	92.5 ± 1.5	$\mathbf{62.3 \pm 2.0}^{**}$	$\mathbf{58.7 \pm 7.5\%}$	78.1%	11.1%
Backbone: ProtoPFormer (LN + Attention Biases) — Dataset: Stanford Dogs-C					
Unadapted	98.2 ± 0.7	53.5 ± 10.6	$56.6 \pm 12.3\%$	0.0%	100.0%
MEMO	98.2 ± 0.6	55.1 ± 10.9	$57.9 \pm 12.4\%$	100.0%	1.0%
SAR	96.6 ± 0.5	34.6 ± 9.4	$43.4 \pm 12.6\%$	**0.8%**	**89.8%**
Tent	98.2 ± 0.5	57.2 ± 8.6	$61.5 \pm 10.2\%$	100.0%	55.8%
EATA	98.2 ± 0.5	57.2 ± 8.5	$61.5 \pm 10.0\%$	69.9%	55.2%
***ProtoTTA* (adaptive routing)**	$\mathbf{98.3 \pm 0.5}$	$\mathbf{57.5 \pm 8.3}$	$\mathbf{61.7 \pm 9.6\%}$	73.6%	14.9%
Backbone: ProtoLens (LN + Attention Biases) — Dataset: Amazon-C					
Unadapted	72.9 ± 15.0	64.4 ± 3.9	$81.7 \pm 8.6\%$	0.0%	100.0%
SAR	69.1 ± 20.5	64.3 ± 4.4	$79.6 \pm 10.7\%$	14.5%	47.8%
Tent	70.4 ± 17.5	63.9 ± 4.4	$80.6 \pm 9.6\%$	100.0%	92.8%
EATA	72.7 ± 15.1	64.3 ± 3.9	$81.7 \pm 8.6\%$	**2.0%**	**98.6%**
***ProtoTTA* (adaptive routing)**	$\mathbf{74.4 \pm 15.4}^{**}$	$\mathbf{64.8 \pm 4.0}^{**}$	$\mathbf{82.5 \pm 8.8\%}^{**}$	17.1%	66.3%

source-free. Blur remains the hardest category for both Transformer backbones, suggesting sensitivity of fine-grained patch-token matching. Contrast is hardest for ProtoPNet on histopathology, suggesting greater dependence on intensity and texture. ProtoPFormer's largest gains occur under noise and blur, with smaller improvements for digital corruptions. Cross-domain ProtoLens (NLP domain) results further confirm that the approach generalizes beyond vision despite its shared, non-class-specific prototypes.

Role of the Interpolation Coefficient. The fixed-λ sweep (Table 4) shows that the balance between prototype and output entropy depends on the backbone, consistent with Proposition 2. ProtoViT favors prototype-only adaptation; ProtoLens performs similarly with prototype-only and adaptive weighting, which suggests that shared prototypes also provide a useful adaptation signal. ProtoPNet and ProtoPFormer depend more on output entropy: ProtoPNet favors a mixture, while ProtoPFormer gains little over output-only adaptation. Both deteriorate under prototype-only adaptation. The adaptive controller stays close to the best fixed settings without tuning λ per backbone.

Adaptation Changes Reasoning, Not Only Accuracy. Table 2 shows that *ProtoTTA* attains the highest PCA-W and prediction stability on all four backbones, and the highest PAC on three of four settings. On ProtoPNet, its PAC remains within 0.1 point of the best method, which indicates that improved correct-class prototype alignment does not come at the expense of clean–corrupted activation consistency. In several settings, *ProtoTTA*'s interpretability gains exceed its accuracy gains (e.g., ProtoLens PAC: 1.7 vs. 0.5 points over EATA). This suggests that prototype guidance helps recover clean-input evidence patterns alongside better predictions, while accuracy gains alone need not imply the same consistency. Operationally, *ProtoTTA*'s additional gradient computations reduce throughput, especially on ProtoPNet and ProtoPFormer. These results answer RQ2 while making the computational trade-off explicit.

Qualitative Analysis. Under noise corruption, the unadapted model and EATA keep localized or weakened prototype evidence (Figure 3). ProtoTTA recovers distributed spatial evidence and stronger correct-class activations to better restore the correct prediction. This is consistent with Proposition 1: prototype-level supervision can update intermediate representations unchanged by output-level supervision. See Appendix J for more examples and analyses.

Table 3: VLM evaluation on the 100-sample CUB-200-C subset (1–5 scale; left), and Pearson (r) / Spearman (ρ) correlations between exact sample-level PCA-W and VLM scores on a disjoint 650-sample subset (right).

Method	Focus Rel. ↑	Proto. Match ↑	Overall ↑
Unadapted	4.14 ± 0.98	3.66 ± 0.87	3.53 ± 1.02
Tent	4.03 ± 0.92	3.64 ± 0.92	3.59 ± 1.02
EATA	$\underline{4.18 \pm 0.90}$	$\underline{3.79 \pm 0.82}$	$\underline{3.75 \pm 0.94}$
ProtoTTA (Ours)	$\mathbf{4.30 \pm 0.90}$	$\mathbf{3.86 \pm 0.81}$	$\mathbf{3.78 \pm 0.97}$

Correlation	Overall r	Overall ρ	Proto. Match r	Proto. Match ρ	Focus Rel. r	Focus Rel. ρ
All samples	0.35	0.51	0.32	0.51	0.32	0.45
Within each method	0.34	0.48	0.31	0.47	0.31	0.46

Table 4: Left: geometric-filtering and adaptation-parameter ablations on ProtoViT/CUB-200-C. Right: sensitivity to the interpolation coefficient λ across three prototypical backbones.

Configuration	Mean Acc.	Std. Dev.	Min	Max
Geometric Filtering				
No Filter	56.45%	14.77	34.04	74.76
With Filter (Ours)	**60.15%**	**11.13**	**42.32**	**74.92**
Adaptation Parameters				
All Parameters ($lr = 10^{-3}$)	1.32%	0.75	0.74	3.62
Prototype Modules + Head	39.21%	11.17	24.08	60.51
LayerNorm Only	59.43%	11.49	41.15	75.27
All Parameters ($lr = 10^{-5}$)	59.63%	11.52	41.92	75.44
LN + Attn Bias (Ours)	**60.15%**	**11.13**	**42.32**	**74.92**

Setting	ProtoViT CUB-200-C	ProtoPFormer Dogs-C	ProtoPNet SICAPv2-C
$\lambda = 0.0$ (output only)	55.92%	$\underline{57.43\%}$	55.97%
$\lambda = 0.2$	57.68%	**57.47%**	56.02%
$\lambda = 0.4$	59.36%	57.16%	**56.17%**
$\lambda = 0.5$	59.89%	56.99%	$\underline{56.16\%}$
$\lambda = 0.7$	60.22%	56.64%	55.84%
$\lambda = 0.8$	$\underline{60.28\%}$	56.37%	54.36%
$\lambda = 1.0$ (prototype only)	**60.33%**	55.77%	51.70%

VLM-Based Interpretability Evaluation. We pass reasoning boards from a representative CUB-200-C subset to a VLM agent that scores focus relevance, prototype match, and overall adaptation quality (protocol in Appendix E). Table 3 shows that *ProtoTTA* achieves the highest scores across all dimensions, with the largest gains in focus relevance ($+0.12$ over EATA) and prototype match ($+0.20$ over unadapted), providing complementary evidence of improved prototype focus. Furthermore, on 1,300 method–image observations, exact sample-level PCA-W correlates positively with VLM-rated overall quality (Pearson r=0.35, Spearman ρ=0.51; Table 3). The correlation remains similar within each method after removing method-specific averages (r=0.34, ρ=0.48), with consistent associations for prototype match and focus relevance, supporting PCA-W as an interpretability measure aligned with the VLM-based assessment (see Appendix E.3 for the sampling and correlation details). Figure 2 illustrates how ProtoTTA shifts attention from noisy beak/crown artifacts to the discriminative red eye and retrieves correct-class prototypes for the Bronzed Cowbird. The VLM explains this change automatically, answering RQ3. While these automated judgments complement the quantitative metrics, validation by human judgments remains future work. Beyond evaluation, Appendix C demonstrates that these reasoning boards can guide a label-free VLM judge to intervene on individual predictions, further improving *ProtoTTA*'s performance across three of the four backbones (Table 7). For instance, it yields gains of $+1.44$ percentage points on ProtoPFormer.

Ablation Studies. Removing geometric filtering reduces mean accuracy from 60.15% to 56.45% and worst-corruption accuracy from 42.32% to 34.04% (Table 4). Full-model updates collapse at 10^{-3} but recover 59.63% at 10^{-5}; updating normalization and attention biases reaches 60.15%. These results support constrained adaptation while showing that collapse depends on the learning rate. Updating only prototype modules and the head underperforms (39.21%), and LayerNorm alone trails the full configuration (59.43%), showing normalization and attention are best updated jointly. Appendix I reports gains from Top-K Mean over maximum aggregation (60.15% vs. 59.61%), target-only adaptation, and combined confidence/importance weighting.

6 Conclusion

We presented *ProtoTTA*, a test-time adaptation method for inherently interpretable prototype-based models. *ProtoTTA* combines binary entropy minimization on mapped prototype activations with output entropy, using label-free activation and class-evidence margins to select and stabilize the updates. Our analysis shows that prototype-level losses can provide supervision along activation directions to which output-level losses are locally insensitive, which justifies our adaptively mixing the two. Our findings support the potential for evaluating test-time adaptation not only through predictive accuracy but also through reasoning fidelity. Future work should validate VLM-based assessments against human judgments and extend adaptation guided by prototype interpretability to concept-bottleneck and other inherently interpretable architectures.

AI USE STATEMENT

In this work, the text is the manual effort of the authors, with generative AI tools used for minor language revisions and table formatting. Generative AI tools have been used in the implementation of code and in the assistance of mathematical derivations. This usage has been checked by the authors running these tools. Generative AI tools, and in particular the LLM Feedback tool bundled into OpenReview, have been used to provide feedback on this document which the authors have incorporated through their own edits. The authors take responsibility for the content of this work.

REPRODUCIBILITY STATEMENT

We provide our code as supplementary material accompanying our submission to support reproduction of our experiments. Section 3 describes the proposed method, with pseudocode in Appendix G, the shared-prototype extension in Appendix H, and detailed controller specifications, hyperparameters, and optimization settings in Appendix L. Section 4 and Appendix A document the experimental setup, backbones, datasets, and corruption protocols. Section 3.5 and Appendix E describe the interpretability metrics and VLM-based evaluation protocol.

REFERENCES

Aishwarya Agarwal, Srikrishna Karanam, and Vineet Gandhi. Tide: Training locally interpretable domain generalization models enables test-time correction. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 30210–30220, 2025.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-vl technical report, 2025. URL https://arxiv.org/abs/2511.21631.

Arthur Chen, Zuxin Liu, Jianguo Zhang, Akshara Prabhakar, Zhiwei Liu, Shelby Heinecke, Silvio Savarese, Victor Zhong, and Caiming Xiong. Grounded test-time adaptation for llm agents. *arXiv preprint arXiv:2511.04847*, 2025.

Chaofan Chen, Oscar Li, Daniel Tao, Alina Barnett, Cynthia Rudin, and Jonathan K Su. This looks like that: deep learning for interpretable image recognition. *Advances in neural information processing systems*, 32, 2019.

Jon Donnelly, Alina Jade Barnett, and Chaofan Chen. Deformable protopnet: An interpretable image classifier using deformable prototypes. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10265–10275, 2022.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=YicbFdNTTy.

Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *Proceedings of the International Conference on Learning Representations*, 2019.

Aditya Khosla, Nityananda Jayadevaprakash, Bangpeng Yao, and Fei-Fei Li. Novel dataset for fine-grained image categorization: Stanford dogs. In *Proc. CVPR workshop on fine-grained visual categorization (FGVC)*, volume 2, 2011.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. URL https://arxiv.org/abs/1412.6980.

Jian Liang, Ran He, and Tieniu Tan. A comprehensive survey on test-time adaptation under distribution shifts. *International Journal of Computer Vision*, 133(1):31–64, 2025.

Chiyu Ma, Jon Donnelly, Wenjun Liu, Soroush Vosoughi, Cynthia Rudin, and Chaofan Chen. Interpretable image classification with adaptive prototype-based vision transformers. *Advances in Neural Information Processing Systems*, 37:41447–41493, 2024.

Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Yaofo Chen, Shijian Zheng, Peilin Zhao, and Mingkui Tan. Efficient test-time model adaptation without forgetting. In *The Internetional Conference on Machine Learning*, 2022.

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 *Internetional Conference on Learning Representations*, 2023.

Mihir Prabhudesai, Tsung-Wei Ke, Alex Li, Deepak Pathak, and Katerina Fragkiadaki. Diffusiontta: Test-time adaptation of discriminative models via generative feedback. *Advances in Neural Information Processing Systems*, 36:17567–17583, 2023.

Barbara Rychalska, Dominika Basaj, Alicja Gosiewska, and Przemysław Biecek. Models in the wild: On corruption robustness of neural nlp systems. In *International Conference on Neural Information Processing*, pp. 235–247. Springer, 2019.

Sahil Sethi, David Chen, Thomas Statchen, Michael C Burkhart, Nipun Bhandari, Bashar Ramadan, and Brett Beaulieu-Jones. Protoecgnet: Case-based interpretable deep learning for multi-label ecg classification with contrastive learning. *arXiv preprint arXiv:2504.08713*, 2025.

Julio Silva-Rodríguez. SICAPv2 - prostate whole slide images with gleason grades annotations, 2020. URL https://data.mendeley.com/datasets/9xxm58dvs3/1.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. Mpnet: Masked and permuted pre-training for language understanding. *Advances in neural information processing systems*, 33: 16857–16867, 2020.

Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *International conference on machine learning*, pp. 9229–9248. PMLR, 2020.

Hooman Vaseli, Ang Nan Gu, S Neda Ahmadi Amiri, Michael Y Tsang, Andrea Fung, Nima Kondori, Armin Saadat, Purang Abolmaesumi, and Teresa SM Tsang. Protoasnet: Dynamic prototypes for inherently interpretable and uncertainty-aware aortic stenosis classification in echocardiography. In *International conference on medical image computing and computer-assisted intervention*, pp. 368–378. Springer, 2023.

Hooman Vaseli, Victoria Wu, Nima Kondori, Nguyen Nhat Minh To, Andrea Fung, Ang Nan Gu, and Purang Abolmaesumi. Happi: Hyperbolic hierarchical part prototypes for image recognition. In *2025 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, pp. 696–705. IEEE, 2025.

Guillaume Vray, Devavrat Tomar, Xufeng Gao, Jean-Philippe Thiran, Evan Shelhamer, and Behzad Bozorgtabar. Reservoirtta: Prolonged test-time adaptation for evolving and recurring domains. *arXiv preprint arXiv:2505.14511*, 2025.

Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. Jul 2011.

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.

Bowen Wei and Ziwei Zhu. ProtoLens: Advancing prototype learning for fine-grained interpretability in text classification. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 4503–4523, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.226. URL https://aclanthology.org/2025.acl-long.226/.

Yuanyuan Wei, Roger Tam, and Xiaoying Tang. Mprotonet: A case-based interpretable model for brain tumor classification with 3d multi-parametric magnetic resonance imaging. In *Medical Imaging with Deep Learning*, pp. 1798–1812. PMLR, 2024.

Mengqi Xue, Qihan Huang, Haofei Zhang, Jingwen Hu, Jie Song, Mingli Song, and Canghong Jin. Protopformer: concentrating on prototypical parts in vision transformers for interpretable image recognition. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, pp. 1516–1524, 2024.

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.

A Dataset and Experimental Details

Vision Benchmarks and Models. We evaluate ProtoTTA on three challenging vision datasets that require fine-grained discrimination. For *CUB-200* (Wah et al., 2011), a fine-grained bird classification dataset (200 classes), we use the ProtoViT (Ma et al., 2024) model with a DeiT-S/16 backbone. This model contains 2,000 prototypes (10 per class, with 4 sub-prototypes each) and achieves 86.1% accuracy on clean test data. We construct the *CUB-200-C* dataset using 13 of the corruption types with severity 5, following ImageNet-C (Hendrycks & Dietterich, 2019), categorized as follows:

- **Noise:** Gaussian, Shot, Impulse, Speckle
- **Blur:** Gaussian, Defocus
- **Weather:** Fog, Frost, Brightness
- **Digital:** JPEG Compression, Contrast, Pixelate, Elastic Transform

For *SICAPv2* (Silva-Rodríguez, 2020), a challenging histopathology dataset for prostate cancer grading that requires distinguishing subtle morphological differences between cancer grades, we employ the legacy ProtoPNet architecture (Chen et al., 2019) with a VGG19-BN (Simonyan & Zisserman, 2014) backbone. This model is configured with 50 prototypes (10 per class) and achieves 63.4% accuracy on clean test data. To bridge the gap between ProtoPNet's distance-based metric and the probabilistic requirements of our adaptation method, we apply a specific transformation (a form of Log-Inverse Distance Kernel) to the prototype activations. ProtoPNet natively outputs minimum squared Euclidean distances, d_{min}. We map these to a probability distribution by first computing a raw similarity score s_{raw} via logarithmic activation:

$$s_{raw} = \log\left(\frac{d_{min} + 1.0}{d_{min} + 10^{-4}}\right) \tag{9}$$

which is subsequently normalized to the $[0, 1]$ range. We construct *SICAPv2-C* following the same corruption protocol as CUB-200-C.

For *Stanford Dogs* (Khosla et al., 2011), a fine-grained dog breed classification dataset (120 classes), we use the ProtoPFormer (Xue et al., 2024) model with a DeiT-S/16 backbone. ProtoPFormer extends prototype learning to Vision Transformers through a *token reservation* mechanism: at the last transformer block, only the top-k most-attended patch tokens are retained for prototype comparison, concentrating the model's attention on discriminative image regions. The model employs a dual-branch architecture with 1,800 prototypes in total, 1,200 local patch-level prototypes (10 per class), and 600 global CLS-token prototypes (5 per class), each of dimension 384, with predictions formed as an equal-weight combination of both branches. It achieves 82.4% accuracy on clean test data. We construct *Stanford Dogs-C* using the same 13 corruption types at severity 5 following ImageNet-C (Hendrycks & Dietterich, 2019), with the same category groupings as CUB-200-C.

NLP Benchmark and Model. We evaluate ProtoLens (Wei & Zhu, 2025) for review sentiment classification. We use a ProtoLens model with an *all-mpnet-base-v2* (Song et al., 2020) backbone, configured with 50 shared prototypes. Each prototype represents a concept and is associated with some representative sub-sentences extracted from the training set via a 5-word sliding window for interpretability. The model was trained on the Yelp dataset (94.0% accuracy) and achieves 91.9% accuracy on the clean Amazon test dataset. We create the *Amazon-C* benchmark following WildNLP (Rychalska et al., 2019) with 5 corruption types applied across 4 severity levels (20%, 40%, 60%, and 80% word-level corruption), resulting in 20 total experimental settings. The corruptions are categorized as follows:

- **Keyboard:** QWERTY
- **Character:** Swap, Remove Char
- **Combined:** Mixed, Aggressive

During test-time adaptation, we apply a temperature-scaled sigmoid function to the prototype similarities to map them into the $[0, 1]$ range with an appropriate spread:

$$p_i = \sigma(\tau \cdot s_i) = \frac{1}{1 + e^{-\tau \cdot s_i}} \tag{10}$$

where s_i is the cosine similarity for prototype i, and τ is a temperature hyperparameter (typically set to 5.0) that controls the sharpness of the resulting probability distribution.

A.1 Detailed Corruption Results

Table 5: Test-time adaptation under visual corruptions across different backbones and datasets (accuracy (%) at severity level 5). Each corruption entry averages three seeds. For each ProtoTTA variant, * and ** indicate Holm-adjusted $p < 0.05$ and $p < 0.01$ against the strongest baseline (two-sided paired t-test across corruptions).

	Noise					Blur			Weather				Digital					
Method	Gaussian	Shot	Impulse	Speckle	**Avg**	Defocus	Gaussian	**Avg**	Frost	Fog	Brightness	**Avg**	Contrast	Elastic	JPEG	Pixelate	**Avg**	**Total**
Backbone: ProtoViT (LN + Attention Biases) — Dataset: CUB-200-C																		
Unadapted	37.2	36.5	39.6	48.8	40.5	42.8	39.0	40.9	43.4	61.6	69.2	58.1	49.6	71.0	67.7	68.1	64.1	51.9 ± 13.6
MEMO	37.2	36.7	38.5	48.5	40.2	43.0	39.0	41.0	43.3	63.0	69.6	58.6	53.5	71.7	68.3	69.5	65.7	52.4 ± 14.0
SAR	40.0	38.4	41.2	49.3	42.2	43.0	38.5	40.7	45.2	62.7	70.1	59.3	46.9	71.4	67.7	68.3	63.6	52.5 ± 13.3
Tent	45.8	41.2	40.2	45.5	43.2	41.5	39.2	40.4	49.9	63.0	72.2	61.7	52.9	73.8	67.0	70.2	66.0	54.0 ± 13.3
EATA	50.1	51.4	51.9	61.4	53.7	45.1	41.6	43.3	52.8	69.6	73.3	65.2	55.7	74.8	67.9	70.9	67.3	59.0 ± 11.3
***ProtoTTA* (oracle fixed λ=1.0)**	**52.4**	53.0	**53.9**	**65.1**	**56.1**	**46.4**	**43.2**	**44.8**	53.3	**70.5**	**74.2**	**66.0**	**56.6**	**75.1**	68.0	72.5	**68.0**	**60.3** ± **11.0****
***ProtoTTA* (adaptive routing)**	52.2	**53.1**	53.8	64.9	56.0	46.2	42.3	44.3	**53.3**	70.1	74.2	65.9	55.9	74.9	**68.4**	**72.5**	67.9	60.1 ± 11.1**
Backbone: ProtoPNet (BN + 1×1 Convs) — Dataset: SICAPv2-C																		
Unadapted	16.4	11.1	12.7	11.5	12.9	36.9	34.4	35.6	31.2	30.3	31.9	31.1	30.9	55.8	49.4	55.5	47.9	31.4 ± 15.6
MEMO	58.0	**58.5**	**59.9**	58.4	**58.7**	52.0	55.3	53.6	59.5	52.2	59.3	57.0	**44.6**	57.2	51.6	60.3	53.4	55.9 ± 4.6
SAR	**59.3**	57.7	58.7	57.5	58.3	52.8	55.4	54.1	59.2	51.6	59.5	56.7	42.9	57.8	51.8	61.7	53.5	55.8 ± 5.0
Tent	57.1	55.5	56.1	54.5	55.8	53.1	55.1	54.1	58.5	49.9	54.9	54.4	40.5	58.4	52.3	60.5	52.9	54.3 ± 5.0
EATA	58.6	57.3	59.2	57.3	58.1	52.8	55.1	54.0	59.1	51.1	59.3	56.5	43.3	57.7	52.0	61.3	53.6	55.7 ± 4.9
***ProtoTTA* (oracle fixed λ=0.4)**	58.1	58.0	59.2	58.1	58.3	53.4	54.9	54.2	**59.8**	**52.7**	60.0	**57.5**	43.7	**58.7**	52.5	61.1	54.0	**56.2** ± **4.7**
***ProtoTTA* (adaptive routing)**	58.3	58.3	59.0	**58.5**	58.5	**53.7**	**55.5**	**54.6**	59.6	51.5	**60.1**	57.1	42.8	58.1	**52.9**	**62.0**	**54.0**	**56.2** ± **5.0**
Backbone: ProtoPFormer (LN + Attention Biases) — Dataset: Stanford Dogs-C																		
Unadapted	46.6	46.2	48.1	55.6	49.1	40.4	37.4	38.9	48.0	51.2	69.6	56.3	46.5	57.8	68.5	70.8	60.9	52.8 ± 11.0
MEMO	47.0	46.3	49.0	56.4	49.7	41.2	37.9	39.6	48.8	53.2	**70.9**	57.6	**50.7**	59.4	**69.8**	**72.2**	**63.1**	54.1 ± 11.2
SAR	35.4	32.1	34.0	40.9	35.6	30.6	27.6	29.1	38.4	31.7	60.2	43.4	37.0	50.1	54.6	61.0	50.7	41.0 ± 11.5
Tent	52.7	53.2	54.0	60.9	55.2	46.2	42.0	44.1	53.4	60.5	68.7	60.9	48.7	65.6	67.2	70.4	63.0	57.2 ± 9.1
EATA	52.4	53.0	54.1	61.1	55.1	46.4	43.4	44.9	53.3	59.9	68.8	60.6	48.5	65.2	67.1	70.5	62.8	57.2 ± 8.9
***ProtoTTA* (oracle fixed λ=0.2)**	53.2	**54.3**	**54.6**	**61.6**	**55.9**	46.5	43.9	45.2	53.8	**61.4**	67.9	**61.0**	46.8	**66.2**	66.7	70.2	62.5	**57.5** ± **8.8**
***ProtoTTA* (adaptive routing)**	**53.3**	54.0	54.1	61.5	55.7	**46.5**	**44.1**	**45.3**	**53.9**	61.3	67.7	60.9	47.3	66.1	66.7	69.8	62.5	57.4 ± 8.7

Table 6: Test-time adaptation on Amazon-C using Yelp-trained ProtoLens (accuracy (%) across 5 corruptions and 4 severities $\in \{20, 40, 60, 80\}$). Each corruption–severity entry averages three seeds. For each ProtoTTA variant, * and ** indicate Holm-adjusted $p < 0.05$ and $p < 0.01$ against the strongest baseline.

	Qwerty					Swap					Remove					Mixed					Aggressive					
Method	20	40	60	80	**Avg**	20	40	60	80	**Avg**	20	40	60	80	**Avg**	20	40	60	80	**Avg**	20	40	60	80	**Avg**	**Total**
Unadapted	89.4	85.4	80.0	71.5	81.6	89.8	84.4	76.2	70.6	80.3	90.5	86.8	82.5	**76.8**	84.2	84.1	84.1	84.1	80.3	83.2	87.5	79.5	71.2	61.3	74.9	80.81 ± 7.62
SAR	88.7	85.7	76.7	66.5	79.4	88.7	82.8	69.8	63.9	76.3	88.7	87.1	80.3	70.8	81.7	82.8	82.8	82.8	77.3	81.4	86.9	76.6	65.8	55.2	71.1	77.98 ± 9.69
Tent	**89.5**	85.2	78.0	68.5	80.3	**90.2**	84.0	75.3	68.9	79.6	**90.8**	86.8	81.4	75.1	83.5	83.5	83.5	83.5	79.0	82.4	87.5	77.9	67.8	58.4	72.9	79.74 ± 8.65
EATA	89.4	85.4	79.8	71.5	81.5	89.8	84.2	76.3	**70.7**	80.2	90.6	86.8	82.5	76.8	84.2	84.1	84.1	84.1	80.2	83.1	87.5	79.5	71.0	61.6	74.9	80.79 ± 7.59
***ProtoTTA* (oracle fixed λ=1.0)**	89.4	**86.4**	**81.2**	**72.3**	**82.3**	89.8	85.0	76.1	69.1	80.0	90.3	87.8	**83.4**	75.8	84.4	84.9	84.9	84.9	**81.3**	84.0	**88.3**	80.6	**72.2**	**62.3**	**75.9**	**81.32** ± **7.76***
***ProtoTTA* (adaptive routing)**	89.4	86.3	81.1	71.9	82.2	89.9	**85.0**	**76.6**	70.1	**80.4**	90.3	**87.9**	83.3	76.4	**84.5**	**85.0**	**85.0**	**85.0**	81.3	**84.1**	88.2	**80.7**	71.8	61.1	75.4	81.31 ± 7.79**

B Applicability Beyond Vision

ProtoTTA operates on prototype similarities and predictive distributions rather than raw inputs. It can therefore be instantiated for other modalities when a pretrained prototype-based model exposes differentiable similarity scores and suitable test-time adaptation parameters. Figure 4 illustrates text classification: encoded text spans are compared with learned prototypes represented by exemplar phrases. Reliable-sample filtering and adaptive interpolation follow the same framework, with backbone-specific similarity mappings and thresholds. For prototypes shared across classes, classifier-derived importance weights and soft targets replace hard class membership, yielding an importance-weighted binary cross-entropy prototype objective. Prototypes and classifier weights remain fixed; consensus aggregation is optional when sub-prototypes are available.

C Interpretability-Guided Judge Intervention

We conduct a separate intervention experiment using `Qwen/Qwen3.6-35B-A3B` model. For samples whose pre- and post-adaptation predictions disagree, the judge receives the corrupted input together with the corresponding before/after prediction evidence and prototype reasoning. If the evidence indicates harmful adaptation, the post-adaptation prediction is rejected and replaced by the pre-adaptation prediction; the model parameters themselves are not rolled back. For ProtoLens, the same protocol uses the corrupted text and retrieved prototype snippets.

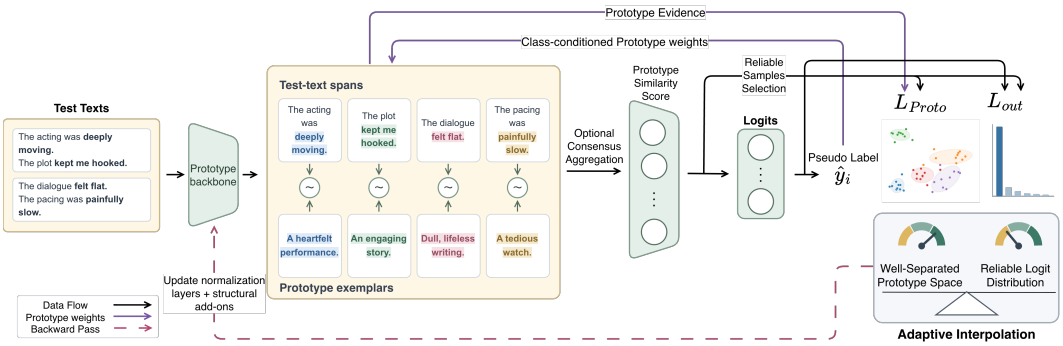


Figure 4: ProtoTTA for text classification. Matching highlights indicate illustrative span–prototype correspondences; $\sim$ denotes feature-space similarity. The adaptation pipeline is retained, with class-conditioned weighting for shared prototypes.

Table 7: Label-free judge-guided inference across three seeds. Accuracy is the macro-average over corruption settings, and SD is computed across seed-level macro-averages. W/T/L counts settings in which the judge improves, ties, or degrades *ProtoTTA*.

Backbone	**Benchmark**	***ProtoTTA***	**+ Judge**	**Δ (pp)**	**W/T/L**
ProtoViT	CUB-200-C	60.368 ± 0.025	60.487 ± 0.030	$+\mathbf{0.119}$	10/0/3
ProtoPNet	SICAPv2-C	56.18 ± 0.29	51.96 ± 0.12	-4.22	0/0/13
ProtoPFormer	Stanford Dogs-C	57.444 ± 0.052	58.882 ± 0.045	$+\mathbf{1.438}$	13/0/0
ProtoLens	Amazon-C	81.311 ± 0.000	82.345 ± 0.008	$+\mathbf{1.034}$	20/0/0

Overall, these results indicate that exposing prototype-level reasoning to an external judge can serve as an effective, label-free supervision signal for test-time updates, improving accuracy in three of four settings, with gains of up to 1.4 points. The exception is ProtoPNet, where judge guidance instead degrades accuracy. We attribute this to properties of the backbone rather than a general limitation of the approach: ProtoPNet is among the earliest prototypical architectures, and unlike ProtoViT and ProtoPFormer, it lacks sub-prototypes, relying instead on coarser, less fine-grained prototype representations that offer the judge weaker and less spatially localized evidence to reason over. This suggests that the reliability of judge-guided intervention depends on the granularity of the underlying prototype evidence, and that older, non-decomposed architectures may require judges specifically adapted to their coarser representations. We view this experiment as a proof of concept rather than a deployable safeguard: it demonstrates that the interpretability inherent to prototype-based models can support human-in-the-loop or AI-in-the-loop oversight of test-time adaptation, an option unavailable to black-box TTA methods, while showing that this benefit is not yet uniform across backbones.

D Additional Theoretical Analysis and Proofs

D.1 Local gradient analysis

We provide a local gradient interpretation of the two signals in Equation 3. For a test sample x_i, let $\boldsymbol{s}_i \in \mathbb{R}^P$ denote the classifier-facing prototype activations after any spatial or sub-prototype aggregation, and let

$$\boldsymbol{z}_i = h(\boldsymbol{s}_i) \in \mathbb{R}^C, \qquad \boldsymbol{p}_i = \mathrm{softmax}(\boldsymbol{z}_i).$$

The prototype loss uses mapped activations $\bar{s}_{ip} = \phi_p(s_{ip}) \in (0, 1)$. We analyze a local region where the mappings are differentiable with $\phi'_p(s_{ip}) > 0$ and numerical clipping is inactive. Let $\mathcal{A}_i = \mathcal{P}_{\hat{y}_i}$ be the prototypes selected by the pseudo-label and let $w_{ip} > 0$ denote their importance weights. The pseudo-label, selected set, and weights are held fixed during differentiation.

Proposition 1 (Output compression and prototype-level supervision)**.** *For any differentiable output-level objective*

$$\ell_i^{\text{out}} = F_i(h(\boldsymbol{s}_i)),$$

let $\boldsymbol{r}_i = \nabla_{\boldsymbol{z}_i} F_i(\boldsymbol{z}_i)$ *and* $\boldsymbol{J}_{h,i} = \partial h(\boldsymbol{s}_i)/\partial \boldsymbol{s}_i$*. Then*

$$\nabla_{\boldsymbol{s}_i} \ell_i^{\text{out}} = \boldsymbol{J}_{h,i}^\top \boldsymbol{r}_i \in \text{range}(\boldsymbol{J}_{h,i}^\top),$$

whose dimension is at most C*. Hence, for every* $\boldsymbol{d} \in \ker(\boldsymbol{J}_{h,i})$*,*

$$D_{\boldsymbol{d}} \ell_i^{\text{out}} = 0.$$

An output-level loss is therefore locally insensitive to prototype changes that leave the logits unchanged to first order.

In contrast, the prototype loss

$$\ell_i^{\text{proto}} = \sum_{p \in \mathcal{A}_i} w_{ip} H_{\text{b}}(\bar{s}_{ip})$$

has coordinate derivatives

$$\frac{\partial \ell_i^{\text{proto}}}{\partial s_{ip}} = w_{ip} \phi_p'(s_{ip}) \log \frac{1 - \bar{s}_{ip}}{\bar{s}_{ip}}, \qquad p \in \mathcal{A}_i.$$

Therefore, its coordinate-level directions are determined separately by the individual prototype activations. Unlike the output gradient, the prototype gradient is not forced by the chain rule to lie in $\text{range}(\boldsymbol{J}_{h,i}^\top)$*. Whenever it has a nonzero component in* $\ker(\boldsymbol{J}_{h,i})$*, it supplies first-order supervision along a direction to which every output-level objective is locally insensitive.*

For a class-exclusive positive linear head, $W_{cp} = v_p \mathbb{1}[c = c(p)]$ with $v_p > 0$, the output derivative is $v_p r_{i,c(p)}$. All prototypes belonging to the same class therefore receive output-level coordinate gradients with the same sign. The prototype loss can instead apply opposite pressures within that class: under gradient descent, activations above $\frac{1}{2}$ are pushed upward and those below $\frac{1}{2}$ downward. Proposition 1 characterizes the supervision at the prototype interface; the actual parameter gradient is obtained by multiplying it by the parameter-to-activation Jacobian. The proof appears in Section Appendix D.2.

The activation margin also has a direct interpretation. Since

$$H_{\text{b}}'(q) = \log \frac{1 - q}{q}$$

changes sign only at $q = \frac{1}{2}$, the quantity $|\bar{s}_{ip} - \frac{1}{2}|$ is the smallest perturbation that can move the activation across the gradient-sign boundary. It therefore measures local directional stability, motivating λ_i^{act}. Because a large margin does not by itself guarantee class-discriminative evidence, we complement it with λ_i^{rel}. This $\frac{1}{2}$-boundary interpretation applies to the binary-entropy objective. For ProtoLens's soft-target cross-entropy objective, the corresponding sign boundary is $\bar{s}_{ip} = t_{ip}$ for positively weighted, unclipped coordinates. Its $\frac{1}{2}$-based controller margin is therefore a heuristic measure of activation decisiveness, rather than the distance to its loss-gradient sign boundary.

The preceding analysis explains why prototype entropy supplies fine-grained supervision and why the activation margin measures the local stability of that signal. We next explain why the prototype and output signals should be combined adaptively. Define the normalized parameter-space gradients used by Equation 3:

$$\widetilde{\boldsymbol{g}}_{P,i} = \frac{\nabla_\theta \ell_i^{\text{proto}}}{g_{\text{proto}} + \epsilon}, \qquad \widetilde{\boldsymbol{g}}_{L,i} = \frac{\nabla_\theta \ell_i^{\text{out}}}{g_{\text{out}} + \epsilon}.$$

Let $\boldsymbol{u}_i$ be an ideal local target gradient, used only for analysis, and define errors $\boldsymbol{e}_{P,i} = \widetilde{\boldsymbol{g}}_{P,i} - \boldsymbol{u}_i$ and $\boldsymbol{e}_{L,i} = \widetilde{\boldsymbol{g}}_{L,i} - \boldsymbol{u}_i$.

Proposition 2 (State-dependent optimal interpolation)**.** *Let*

$$A_i = \mathbb{E}\|\boldsymbol{e}_{P,i}\|_2^2, \quad B_i = \mathbb{E}\|\boldsymbol{e}_{L,i}\|_2^2, \quad C_i = \mathbb{E}[\boldsymbol{e}_{P,i}^\top \boldsymbol{e}_{L,i}].$$

If $A_i + B_i - 2C_i > 0$*, the expected squared error of*

$$\lambda \widetilde{\boldsymbol{g}}_{P,i} + (1 - \lambda) \widetilde{\boldsymbol{g}}_{L,i}$$

is uniquely minimized over $\lambda \in [0,1]$ *by*

$$\lambda_i^\star = \mathrm{clip}\left(\frac{B_i - C_i}{A_i + B_i - 2C_i},\, 0,\, 1\right). \tag{11}$$

When $C_i = 0 \ldots$, *this reduces to* $\lambda_i^\star = B_i/(A_i + B_i)$.

The optimal prototype weight therefore increases when the prototype signal is more reliable or the output signal has larger error, and decreases when prototype selection is noisy. Since these errors are unobservable in TTA, our controller uses activation margin, relative class evidence, and half-batch gradient consistency as observable proxies. These proxies motivate the adaptive interpolation; they are not assumed to recover the oracle coefficient exactly.

D.2 Proofs and implications

This section proves Propositions 1 and 2 and clarifies their connection to the practical controller.

Proof of Proposition 1. For a fixed sample i, write

$$\boldsymbol{z}_i = h(\boldsymbol{s}_i), \qquad \boldsymbol{r}_i = \nabla_{\boldsymbol{z}_i} F_i(\boldsymbol{z}_i), \qquad \boldsymbol{J}_{h,i} = \frac{\partial h(\boldsymbol{s}_i)}{\partial \boldsymbol{s}_i}.$$

The chain rule gives

$$\nabla_{\boldsymbol{s}_i} \ell_i^{\mathrm{out}} = \boldsymbol{J}_{h,i}^\top \boldsymbol{r}_i.$$

Therefore,

$$\nabla_{\boldsymbol{s}_i} \ell_i^{\mathrm{out}} \in \mathrm{range}(\boldsymbol{J}_{h,i}^\top),$$

and

$$\dim \mathrm{range}(\boldsymbol{J}_{h,i}^\top) = \mathrm{rank}(\boldsymbol{J}_{h,i}) \le C.$$

For any $\boldsymbol{d} \in \ker(\boldsymbol{J}_{h,i})$, the directional derivative is

$$D_{\boldsymbol{d}} \ell_i^{\mathrm{out}} = \left(\nabla_{\boldsymbol{s}_i} \ell_i^{\mathrm{out}}\right)^\top \boldsymbol{d} = \boldsymbol{r}_i^\top \boldsymbol{J}_{h,i} \boldsymbol{d} = 0.$$

Thus, an output-level objective is locally insensitive to every prototype-activation direction that leaves the logits unchanged to first order.

For the prototype objective,

$$\ell_i^{\mathrm{proto}} = \sum_{p \in \mathcal{A}_i} w_{ip} H_{\mathrm{b}}(\phi_p(s_{ip})),$$

where

$$H_{\mathrm{b}}(q) = -q \log q - (1-q)\log(1-q).$$

Direct differentiation gives

$$H_{\mathrm{b}}'(q) = \log \frac{1-q}{q},$$

and hence

$$\frac{\partial \ell_i^{\mathrm{proto}}}{\partial s_{ip}} = w_{ip} \phi_p'(s_{ip}) \log \frac{1 - \bar{s}_{ip}}{\bar{s}_{ip}}, \qquad p \in \mathcal{A}_i.$$

The derivative is zero for unselected prototypes. Because $w_{ip} > 0$ and $\phi_p'(s_{ip}) > 0$, its sign is positive when $\bar{s}_{ip} < \frac{1}{2}$, negative when $\bar{s}_{ip} > \frac{1}{2}$, and zero at $\bar{s}_{ip} = \frac{1}{2}$. These coordinate signs are determined separately rather than through the output Jacobian.

The prototype gradient is not guaranteed to lie outside $\mathrm{range}(\boldsymbol{J}_{h,i}^\top)$ for every sample. Nevertheless, if there exists $\boldsymbol{d} \in \ker(\boldsymbol{J}_{h,i})$ such that

$$\boldsymbol{d}^\top \nabla_{\boldsymbol{s}_i} \ell_i^{\mathrm{proto}} \neq 0,$$

then the prototype loss supplies a first-order signal along a direction to which every output-level objective is locally insensitive.

For the special case

$$h(\boldsymbol{s}_i) = \boldsymbol{W}\boldsymbol{s}_i + \boldsymbol{b}, \qquad W_{cp} = v_p \mathbb{1}[c = c(p)], \qquad v_p > 0,$$

we obtain

$$\frac{\partial \ell_i^{\text{out}}}{\partial s_{ip}} = (\boldsymbol{W}^\top \boldsymbol{r}_i)_p = v_p r_{i,c(p)}.$$

All prototypes assigned to the same class therefore receive coordinate-level output gradients with the same sign, or all receive zero gradient when $r_{i,c(p)} = 0$. The prototype loss can assign different signs because its derivative depends separately on each $\bar{s}_{ip}$. This proves the proposition. □

Remark 1 (Activation margin as a sign-stability radius)**.** *For $q \in (0,1)$, the sign of $H'_{\text{b}}(q)$ changes only at $q = \frac{1}{2}$. Let*

$$m(q) = \left| q - \frac{1}{2} \right|.$$

For every perturbation Δ satisfying

$$|\Delta| < m(q) \quad \textit{and} \quad q + \Delta \in (0,1),$$

$q + \Delta$ remains on the same side of $\frac{1}{2}$ as q. Therefore,

$$\operatorname{sign} H'_{\text{b}}(q + \Delta) = \operatorname{sign} H'_{\text{b}}(q).$$

The activation margin is consequently the distance to the point where the coordinate-level prototype-gradient direction can change. It is a measure of local directional stability, not a guarantee of correctness or low variance of the parameter gradient.

Remark 2 (Relation to parameter updates)**.** *Let*

$$\boldsymbol{J}_{s_i,\theta} = \frac{\partial \boldsymbol{s}_i}{\partial \theta}.$$

For either component $j \in \{\text{proto}, \text{out}\}$,

$$\nabla_\theta \ell_i^j = \boldsymbol{J}_{\boldsymbol{s}_i,\theta}^\top \nabla_{\boldsymbol{s}_i} \ell_i^j.$$

Proposition 1 characterizes the supervision supplied at the prototype-activation interface. Its realized effect depends on the shared parameter-to-activation Jacobian. For this reason, the controller measures half-batch gradient consistency in the actual parameter space rather than assuming that prototype coordinates are independently adjustable.

Proof of Proposition 2. For a fixed sample i, define

$$\widetilde{\boldsymbol{g}}_i(\lambda) = \lambda \widetilde{\boldsymbol{g}}_{P,i} + (1-\lambda)\widetilde{\boldsymbol{g}}_{L,i}.$$

Since

$$\boldsymbol{e}_{P,i} = \widetilde{\boldsymbol{g}}_{P,i} - \boldsymbol{u}_i, \qquad \boldsymbol{e}_{L,i} = \widetilde{\boldsymbol{g}}_{L,i} - \boldsymbol{u}_i,$$

the mixture error is

$$\widetilde{\boldsymbol{g}}_i(\lambda) - \boldsymbol{u}_i = \lambda \boldsymbol{e}_{P,i} + (1-\lambda)\boldsymbol{e}_{L,i}.$$

Its expected squared norm is

$$M_i(\lambda) = \mathbb{E}\,\|\lambda \boldsymbol{e}_{P,i} + (1-\lambda)\boldsymbol{e}_{L,i}\|_2^2 \tag{12}$$

$$= \lambda^2 A_i + (1-\lambda)^2 B_i + 2\lambda(1-\lambda)C_i. \tag{13}$$

Its second derivative is

$$M_i''(\lambda) = 2(A_i + B_i - 2C_i) = 2\mathbb{E}\,\|\boldsymbol{e}_{P,i} - \boldsymbol{e}_{L,i}\|_2^2.$$

When $A_i + B_i - 2C_i > 0$, the objective is strictly convex. Its first derivative is

$$M_i'(\lambda) = 2\lambda(A_i + B_i - 2C_i) - 2(B_i - C_i).$$

Setting this derivative to zero gives the unconstrained minimizer

$$\lambda_{i,0}^\star = \frac{B_i - C_i}{A_i + B_i - 2C_i}.$$

Restricting the coefficient to $[0, 1]$ gives

$$\lambda_i^\star = \mathrm{clip}\left(\lambda_{i,0}^\star, 0, 1\right).$$

When $C_i = 0$ and $A_i + B_i > 0$, this becomes

$$\lambda_i^\star = \frac{B_i}{A_i + B_i}.$$

If $A_i + B_i - 2C_i = 0$, then

$$\mathbb{E}\,\|\boldsymbol{e}_{P,i} - \boldsymbol{e}_{L,i}\|_2^2 = 0,$$

so the two errors are equal almost surely and every interpolation coefficient gives the same expected error. □

Remark 3 (Bias–variance special case)**.** *Suppose*

$$\widetilde{\boldsymbol{g}}_{P,i} = \boldsymbol{u}_i + \boldsymbol{\varepsilon}_{P,i}, \qquad \widetilde{\boldsymbol{g}}_{L,i} = \boldsymbol{u}_i - \boldsymbol{b}_i + \boldsymbol{\varepsilon}_{L,i},$$

where the noise terms are zero-mean and uncorrelated with each other and with the deterministic output bias $\boldsymbol{b}_i$. If

$$\mathbb{E}\|\boldsymbol{\varepsilon}_{P,i}\|_2^2 = V_{P,i}, \qquad \mathbb{E}\|\boldsymbol{\varepsilon}_{L,i}\|_2^2 = V_{L,i},$$

then

$$A_i = V_{P,i}, \qquad B_i = \|\boldsymbol{b}_i\|_2^2 + V_{L,i}, \qquad C_i = 0.$$

Proposition 2 reduces to

$$\lambda_i^\star = \frac{\|\boldsymbol{b}_i\|_2^2 + V_{L,i}}{\|\boldsymbol{b}_i\|_2^2 + V_{L,i} + V_{P,i}}.$$

This special case shows that the prototype weight increases when the output signal has greater structural bias or noise and decreases when the prototype signal is noisier. Output-space compression from Proposition 1 is one possible source of output bias, but Proposition 2 does not require this particular error model.

Remark 4 (Gradient normalization)**.** *The batch-gradient scales in Equation 2 are treated as constants during the final backward pass. Therefore,*

$$\nabla_\theta \left(\frac{\ell_i^j}{g_j + \epsilon} \right) = \frac{\nabla_\theta \ell_i^j}{g_j + \epsilon}.$$

The unified objective consequently has gradient

$$\nabla_\theta \mathcal{L}_\lambda = \frac{1}{|\mathcal{R}|} \sum_{i \in \mathcal{R}} c_i \left[\lambda_i \widetilde{\boldsymbol{g}}_{P,i} + (1 - \lambda_i) \widetilde{\boldsymbol{g}}_{L,i} \right].$$

The common confidence factor c_i scales the contribution of sample i but does not change its relative interpolation between the two signals. Gradient normalization removes their leading scale disparity; it does not assume that their errors or geometries are identical.

E Interpretability Evaluation Details

E.1 Scoring Protocol

We construct a fixed 100-sample subset of CUB-200-C by sampling approximately uniformly across corruption types at severity 5. For each sample and each TTA method, we assemble a reasoning board comprising: (i) the corrupted test image, (ii) the predicted-class reasoning board showing the top-matched prototypes with their spatial focus regions highlighted, and (iii) the any-class contribution board showing the highest-contributing prototypes across all classes alongside their contribution scores. These boards are passed to a VLM agent (Qwen/Qwen3-VL-32B-Thinking (Bai et al., 2025)) that returns three scores on a 1–5 integer scale:

- **Focus Relevance:** whether the highlighted image region corresponds to a meaningful and species-discriminative part rather than background or noise.

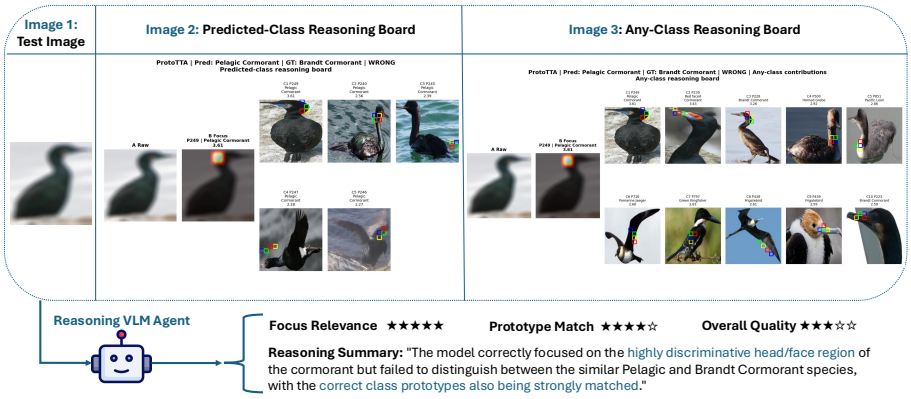


Figure 5: Illustration of the VLM-based scoring protocol. The VLM scores prototype-based reasoning boards to provide language-grounded evaluation of TTA updates unlike black-box methods.

- **Prototype Match:** whether the retrieved prototype patches are visually similar to the highlighted region in the test image.
- **Overall Adaptation Quality:** the overall plausibility of the model's prototype-based reasoning for the predicted class.

Table 3 reports the complete results, including per-metric standard deviations. Furthermore, we measured the overall quality scores exclusively on samples where the models predicted correctly (mean$=4.62$ for ProtoTTA). These filtered scores consistently mirrored the general performance trends, demonstrating that the positive correlation between predictive accuracy and VLM-judged reasoning quality persists across adaptation methods.

E.2 PCA-W Implementation and Shared-Prototype Rule

All PCA-W results in Table 2 use $k = 10$. For ProtoViT, ProtoPNet, and ProtoPFormer, prototypes have fixed class identities and we use Equation 8 directly. ProtoLens is different: its $P = 50$ prototypes are shared across classes, so we do not assign a prototype to a single class, either manually or by taking an argmax over classifier weights.

For a ProtoLens sample i with ground-truth class y_i, let $a_{i,p}$ be the similarity of shared prototype p, let $\mathcal{T}_i$ contain the ten largest similarities, and let $W_{y_i,p}$ be the learned weight from prototype p to class y_i in the frozen classifier head. The implementation first normalizes the selected activations within the top-ten set,

$$\alpha_{i,p} = \frac{\exp(a_{i,p})}{\sum_{q\in\mathcal{T}_i}\exp(a_{i,q})}, \qquad p \in \mathcal{T}_i, \tag{14}$$

and computes the sample-level soft alignment

$$\text{PCA-W}_{\text{PL}}(i) = \frac{\sum_{p\in\mathcal{T}_i}\alpha_{i,p}[W_{y_i,p}]_+}{\sum_{p\in\mathcal{T}_i}\alpha_{i,p}|W_{y_i,p}| + 10^{-8}}, \qquad [u]_+ = \max(u, 0). \tag{15}$$

Thus, positive classifier weights contribute supporting evidence for the ground-truth class, whereas negative weights contribute opposing evidence only to the denominator. This is a soft, class-conditional alignment score and does not imply exclusive prototype membership. The reported ProtoLens PCA-W percentage is $100N^{-1}\sum_{i=1}^{N}\text{PCA-W}_{\text{PL}}(i)$. Ground-truth labels are used only to select the corresponding classifier row during offline evaluation and are never used by the adaptation method.

E.3 Exact PCA-W and VLM Correlation Protocol

Evaluation Set. The correlation analysis uses 650 unique CUB-200-C images, sampled uniformly as 50 images from each of the 13 corruption types at severity 5. This subset is disjoint from the 100-image subset used for the VLM score comparison in Table 3. Each image is evaluated using both the

unadapted model and ProtoTTA, producing 650 observations per method and 1,300 method–image observations in total. For each method–image observation, the VLM receives the corrupted input, the predicted-class reasoning board, and the any-class competing-evidence board. Ground-truth labels and prediction correctness are withheld from the VLM.

Exact Sample-Level PCA-W. For an image x with ground-truth class y, we first aggregate the sub-prototype similarities of prototype p into its activation $a_p(x)$. Let $\mathcal{T}_{10}(x)$ denote the ten most activated prototypes and $W_{y,p}$ the final-layer weight connecting prototype p to class y. The activation–importance contribution is

$$c_p(x, y) = a_p(x)|W_{y,p}|. \tag{16}$$

Exact sample-level PCA-W is then defined as

$$\text{PCA-W}(x) = \frac{\sum_{p\in\mathcal{T}_{10}(x)} c_p(x, y)\mathbb{1}[\text{class}(p) = y]}{\sum_{p\in\mathcal{T}_{10}(x)} c_p(x, y)}. \tag{17}$$

A value of 1 indicates that all top activation–importance mass is assigned to ground-truth-class prototypes, whereas a value of 0 indicates that none is. PCA-W uses the ground-truth class only as an offline evaluation metric; this information is not provided to the VLM.

Correlation Analysis. The pooled analysis correlates PCA-W with the VLM scores across all 1,300 method–image observations. To ensure that the correlation is not caused only by average differences between the unadapted and ProtoTTA models, we additionally perform a within-method analysis. For each method independently, we subtract the method-specific mean from PCA-W and from each VLM score:

$$\widetilde{s}_{i,m} = s_{i,m} - \frac{1}{N_m}\sum_{j=1}^{N_m} s_{j,m}, \tag{18}$$

where $s_{i,m}$ denotes either PCA-W or a VLM score for image i under method m, and $N_m = 650$. We then pool the centered observations and calculate Pearson (r) and Spearman (ρ) correlations. Pearson correlation measures linear agreement, while Spearman correlation measures agreement in sample ranking. As shown in Table 3, the similar pooled and within-method results indicate an individual-sample association between PCA-W and the VLM assessment rather than one caused only by differences between the two model states.

E.4 Additional Evaluation Metrics

We evaluate reasoning consistency using paired clean and corrupted inputs. For each sample i, let $\mathbf{a}_i^{\text{clean}}$ and $\hat{y}_i^{\text{clean}}$ denote the prototype activation vector and prediction of the unadapted model on the clean input. Let $\mathbf{a}_i^{\text{adapted}}$ and $\hat{y}_i^{\text{adapted}}$ denote the corresponding quantities on the corrupted input after adaptation. These clean counterparts are used only for offline evaluation and are unavailable to the adaptation procedure. Let N denote the number of evaluated samples.

Prototype Activation Consistency (PAC). PAC measures the average cosine similarity between the clean and adapted-corrupted prototype activation vectors:

$$\text{PAC }(\%) = \frac{100}{N}\sum_{i=1}^{N} \frac{\mathbf{a}_i^{\text{clean}} \cdot \mathbf{a}_i^{\text{adapted}}}{\|\mathbf{a}_i^{\text{clean}}\|_2\|\mathbf{a}_i^{\text{adapted}}\|_2}. \tag{19}$$

Higher PAC indicates greater preservation of the clean-input activation pattern; it does not by itself establish prediction correctness.

Prediction Stability. Prediction Stability measures agreement between predictions on paired clean and adapted-corrupted inputs:

$$\text{Prediction Stability }(\%) = \frac{100}{N}\sum_{i=1}^{N} \mathbb{1}\left[\hat{y}_i^{\text{adapted}} = \hat{y}_i^{\text{clean}}\right]. \tag{20}$$

Because this metric is not conditioned on correctness, it measures decision consistency rather than accuracy preservation.

Operational Metrics. *Selection Rate* is the percentage of test samples selected for adaptation updates. *Relative Speed* is the adapted model's inference throughput expressed as a percentage of the unadapted model's throughput. Lower selection rates do not necessarily imply higher throughput because adaptation methods incur different computational costs.

F Explainable Test-Time Adaptation via Prototype Reasoning

A key advantage of prototype-based architectures is that test-time adaptation is inherently explainable. Unlike black-box TTA methods that operate on output logits, prototype models expose the full evidence chain: which image regions are attended to, which training prototypes are retrieved, and how their contributions change under adaptation. This allows adaptation dynamics to be narrated in meaningful terms.

Figure 2 illustrates a representative example. The corrupted test image (left) shows a heavily noise-corrupted bird. The unadapted model (right, red) misclassifies it as a Crested Auklet by fixating on noisy beak/crown artifacts — the reasoning board shows wrong-class Crested Auklet prototypes dominating both the predicted-class and any-class boards. After ProtoTTA adaptation (middle, green), the model corrects its prediction to the ground-truth Bronzed Cowbird. The reasoning boards reveal *why*: the model shifts attention to the species-discriminative red eye region and retrieves multiple correct-class Bronzed Cowbird prototypes, while suppressing the spurious wrong-class evidence visible in the unadapted boards.

The VLM agent automatically generates this comparative analysis from the reasoning boards, producing structured output covering: unadapted failure analysis, adaptation success analysis, change impact, and a comparative checklist (attention location, wrong-class suppression, prototype match, part focus). This framework enables language-grounded diagnosis of TTA behaviour that is uniquely possible with prototype-based models.

G ProtoTTA Pseudocode

Algorithm 1 summarizes ProtoTTA. For each test batch, the method selects samples with reliable prototype activations, estimates how much to trust prototype entropy relative to output entropy, and updates only the designated adaptation parameters.

H Shared-Prototype Extension

Unlike the other evaluated architectures, ProtoLens does not assign each prototype exclusively to a single class. Instead, its prototypes are shared across classes and contribute through the learned classifier matrix $\boldsymbol{W}$. We therefore replace hard prototype–class membership with classifier-derived importance weights and soft association targets. The same distinction is used for offline interpretability evaluation: ProtoLens PCA-W follows the signed-weight soft-alignment rule in Equation 15, not a hard class assignment.

Let u_{ip}^{PL} denote the cosine similarity between sample i and shared prototype p. The activation used by the prototype objective is

$$\bar{s}_{ip}^{\mathrm{PL}} = \mathrm{clip}\left(\sigma\big(5\ \mathrm{clip}(u_{ip}^{\mathrm{PL}}, -1, 1)\big), \epsilon_H, 1-\epsilon_H\right), \qquad \epsilon_H = 10^{-6}. \tag{21}$$

Thus, ProtoLens uses a fixed similarity temperature of 5.

Given the pseudo-label $\hat{y}_i = \arg\max_c p_{ic}$, the importance and soft association target of prototype p are

$$a_{ip} = \frac{|W_{\hat{y}_i p}|}{\max_r |W_{\hat{y}_i r}| + \epsilon_W}, \qquad t_{ip} = \sigma(2W_{\hat{y}_i p}), \qquad \epsilon_W = 10^{-6}. \tag{22}$$

The ProtoLens prototype objective averages the importance-weighted binary cross-entropy over all $P = 50$ shared prototypes:

$$\ell_i^{\mathrm{proto}} = \frac{1}{P}\sum_{p=1}^{P} a_{ip}\left[-t_{ip}\log \bar{s}_{ip}^{\mathrm{PL}} - (1-t_{ip})\log(1-\bar{s}_{ip}^{\mathrm{PL}})\right]. \tag{23}$$

Algorithm 1 ProtoTTA: Adaptive Prototype-Guided Test-Time Adaptation

Require: Pre-trained prototype model f_{θ_0}, test stream $\mathcal{D}_{\text{test}}$, backbone-specific filtering configuration, learning rate η
Ensure: Adapted model f_θ

Initialize $\theta \leftarrow \theta_0$
Enable updates only for normalization parameters and structural add-ons
for each batch $\mathcal{B} = \{\boldsymbol{x}_i\}_{i=1}^{B}$ from $\mathcal{D}_{\text{test}}$ **do**
  // Forward pass and prototype processing
  Compute output probabilities $\boldsymbol{p}_i$ and prototype similarities s_{ip}
  Aggregate sub-prototype scores using Top-k Mean, when applicable
  Map and clamp similarities to $\bar{s}_{ip} \in [\epsilon_H, 1 - \epsilon_H]$
  Obtain pseudo-labels $\hat{y}_i \leftarrow \arg\max_c p_{ic}$
  // Backbone-specific reliability filtering
  Compute $m_i \in \{0, 1\}$ for each sample using the backbone-specific filter in Appendix L
  $\mathcal{R} \leftarrow \{i \in \{1, \ldots, B\} : m_i = 1\}$
  if $|\mathcal{R}| > 0$ **then**
    // Prototype and output signals
    for each $i \in \mathcal{R}$ **do**
      Compute confidence weight c_i
      if prototypes are class-associated **then**
        $\mathcal{A}_i \leftarrow \mathcal{P}_{\hat{y}_i}$
        $\ell_i^{\text{proto}} \leftarrow \sum_{p \in \mathcal{A}_i} w_{ip} H_{\text{b}}(\bar{s}_{ip})$
      else
        Compute ℓ_i^{proto} using Equation 23
      end if
      $\ell_i^{\text{out}} \leftarrow H(\boldsymbol{p}_i)$
    end for
    // Adaptive interpolation
    Compute gradient scales g_{proto} and g_{out} ▷ Equation 2
    Compute λ_i^{act} and λ_i^{rel}
    Compute batch-level gate G ▷ Appendix L
    $\lambda_i \leftarrow G\lambda_i^{\text{act}} + (1 - G)\lambda_i^{\text{rel}}, \quad i \in \mathcal{R}$
    // Unified adaptation objective
    $\mathcal{L}_\lambda \leftarrow \frac{1}{|\mathcal{R}|} \sum_{i \in \mathcal{R}} c_i \left[\lambda_i \frac{\ell_i^{\text{proto}}}{g_{\text{proto}} + \epsilon} + (1 - \lambda_i) \frac{\ell_i^{\text{out}}}{g_{\text{out}} + \epsilon} \right]$
    Treat c_i, λ_i, and g_j as fixed with respect to θ
    Update the enabled parameters using Adam on $\nabla_\theta \mathcal{L}_\lambda$ with learning rate η
  end if
end for
return f_θ

ProtoLens does not use sample-confidence weighting, so $c_i = 1$ for both the prototype and output components.

When neither clipping operation in Equation 21 is active, the derivative with respect to the raw cosine similarity is

$$\frac{\partial \ell_i^{\text{proto}}}{\partial u_{ip}^{\text{PL}}} = \frac{5}{P} a_{ip} \left(\bar{s}_{ip}^{\text{PL}} - t_{ip} \right). \tag{24}$$

Thus, each shared prototype receives an independent adaptation signal that moves its activation toward the soft class association encoded by the classifier head. At a clipping boundary, the derivative is determined by the corresponding clipped operation.

For the activation-margin controller, prototypes with positive support for the pseudo-label,

$$\mathcal{P}_{\hat{y}_i}^{+} = \{p : W_{\hat{y}_i p} > 0\}, \tag{25}$$

are eligible. The margin is computed from the three largest mapped activations in this set, or from all available activations if fewer than three positively supporting prototypes exist. If the set is empty, the three largest mapped activations among all shared prototypes are used.

For the relative-evidence controller, the prototypes supporting class c are

$$\mathcal{P}_c^+ = \{p : W_{cp} > 0\}. \tag{26}$$

If $\mathcal{P}_c^+$ is empty, it is replaced by the three prototypes having the largest classifier weights W_{cp} for class c. The class-level prototype evidence is then

$$S_{i,c} = \text{mean}\left(\text{Top3}_{p\in\mathcal{P}_c^+}\, \bar{s}_{ip}^{\text{PL}}\right), \tag{27}$$

where all available prototypes are used if the eligible set contains fewer than three elements. The remaining definitions of the activation-margin coefficient, relative-evidence coefficient, and batch-level routing variable are unchanged and are provided in Appendix L.

I Additional Ablation Studies

Table 8 studies three additional components of ProtoTTA. Top-K Mean aggregation achieves the highest mean accuracy and lowest across-corruption variance, reaching 60.15%, compared with 59.79% for mean aggregation and 59.61% for maximum aggregation. This supports retaining the strongest sub-prototype evidence while avoiding sensitivity to a single maximum response.

Restricting the prototype objective to prototypes associated with the pseudo-label slightly improves mean accuracy over using all prototypes (60.15% vs. 60.08%), while requiring fewer prototype-level loss terms. This supports target-only adaptation as both an efficient and focused formulation.

Among the weighting strategies, confidence weighting alone improves over the unweighted objective (60.06% vs. 59.97%), whereas importance weighting alone gives a similar mean accuracy (59.94%) but the lowest across-corruption standard deviation. Combining confidence and importance weighting achieves the highest mean accuracy, 60.15%, and the highest worst-case accuracy, 42.32%. These results suggest that the two weights are most useful in combination: confidence controls each sample's contribution, whereas importance controls the contribution of individual prototypes.

Table 8: Additional design ablations on ProtoViT/CUB-200-C at severity level 5. Top-K Mean gives the best mean accuracy and lowest variance among the aggregation rules; target-only adaptation slightly improves mean accuracy while using fewer loss terms; and combined confidence and importance weighting gives the best mean accuracy among the weighting strategies.

Component	Configuration	Mean Acc.	Std. Dev.	Min	Max
Consensus	Mean	59.79%	11.34	**42.34**	75.37
	Max	59.61%	11.71	39.20	**75.41**
	Top-K Mean	**60.15%**	**11.13**	42.32	74.92
Target Prototypes	All Prototypes	60.08%	11.20	**42.37**	**74.92**
	Target Only	**60.15%**	**11.13**	42.32	**74.92**
Weighting	No Weighting	59.97%	11.19	41.99	74.89
	Importance Only	59.94%	**11.12**	42.13	74.92
	Confidence Only	60.06%	11.21	42.29	**74.94**
	Both	**60.15%**	11.13	**42.32**	74.92

J Qualitative Analysis

In this section, we leverage the inherent interpretability of prototype-based models to visually diagnose failure modes under distribution shifts and compare the effectiveness of different adaptation

strategies. Unlike black-box models, this architecture allows us to trace the reasoning process by examining which prototypes are activated and how strongly they contribute to the final prediction.

Figure 6 provides a deeper insight into why baseline methods fail. By visualizing the top-activated prototypes for the *incorrect* class predicted by EATA, we observe that the model assigns high activation scores to irrelevant features. This confirms that EATA fails to suppress noise-induced artifacts effectively. Instead, the noise deceives the model into "hallucinating" strong matches with incorrect prototypes. This phenomenon, amplified activations for the wrong class (Figure 6) combined with suppressed activations for the correct class (Figure 3), fully explains the drop in robustness.

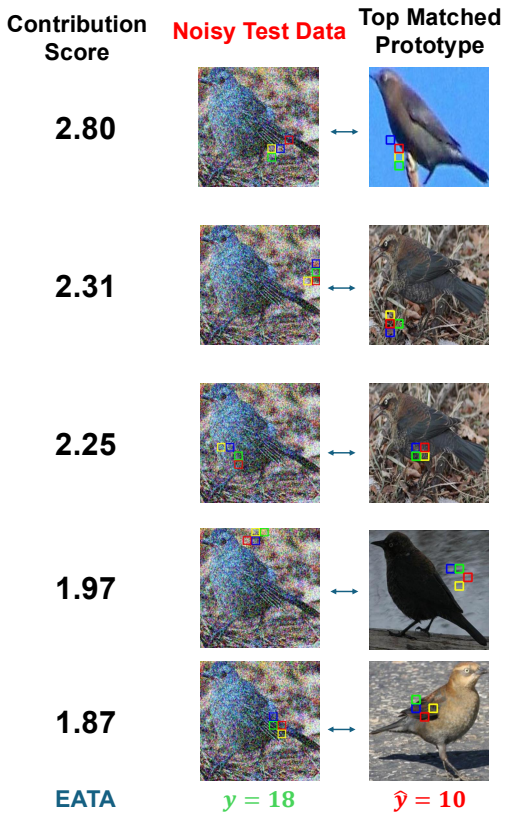


Figure 6: Analysis of EATA Misclassification. We visualize the top-5 activated prototypes for the incorrect class predicted by EATA. Despite the mismatch, the model registers high activation scores (hallucinations), indicating that the adaptation failed to filter out noise-induced features. This contrasts with Figure 3, where ground-truth activations were suppressed.

To further illustrate the contrast in adaptation dynamics, Figure 7 provides a direct comparison of the top-activated prototypes for the predicted class between EATA and ProtoTTA on a corrupted sample (Ground Truth Class: 20). While EATA incorrectly predicts Class 31 by hallucinating strong matches with irrelevant features, ProtoTTA successfully suppresses these noise-induced artifacts. By explicitly minimizing prototype activation entropy , ProtoTTA recovers the correct associations, attending to the appropriate visual features and accurately predicting the ground-truth class.

K Forgetting Analysis

In our primary experimental setup, the model is reset between different corruption types. To investigate the stability of *ProtoTTA* during prolonged adaptation on ProtoViT, we analyze the performance trend over a single long-sequence corruption (Gaussian Noise, Severity 5, approx. 5,500 samples) without intermediate resets. Figure 8 illustrates the per-batch accuracy of *ProtoTTA* compared to EATA and the unadapted baseline. We observe that *ProtoTTA* maintains a consistent performance advantage over the unadapted model throughout the entire sequence. Crucially, there is

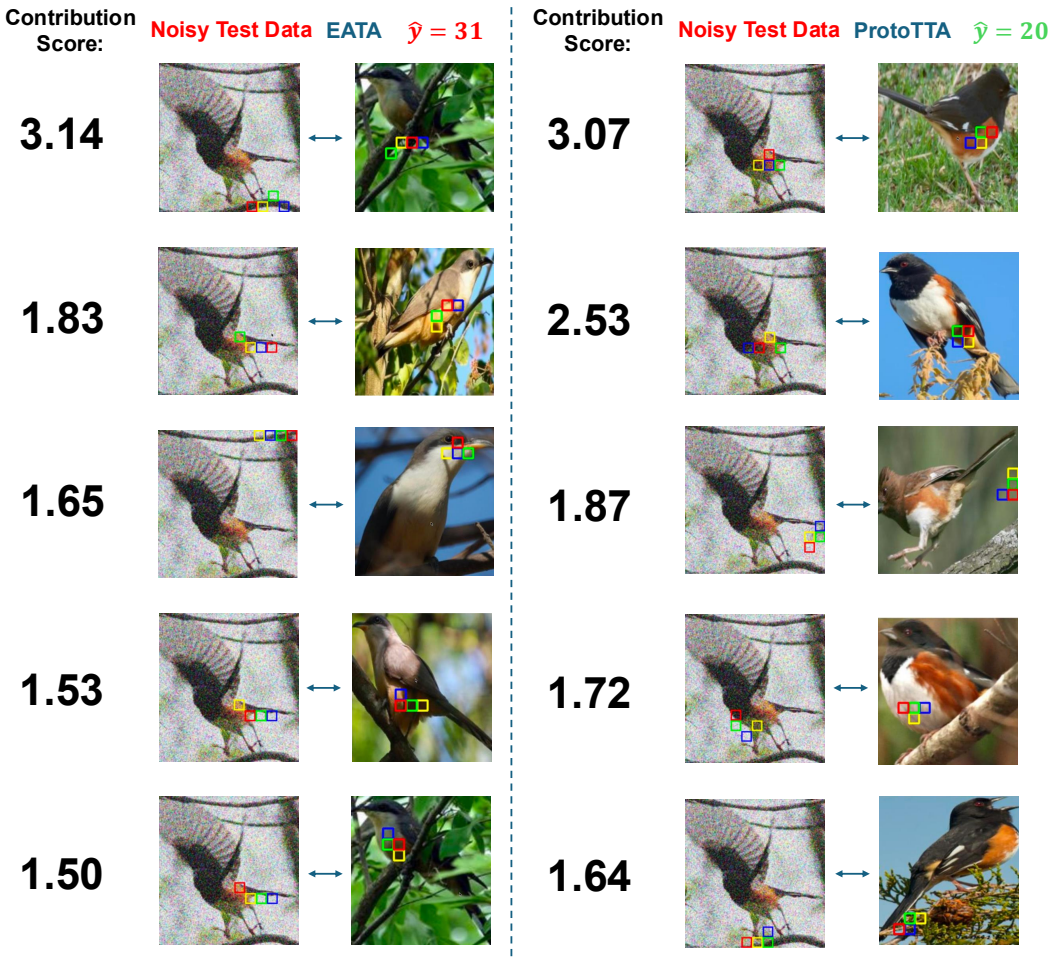


Figure 7: Comparison of Prototype Contributions (Activation $\times$ Weight) for the predicted class between EATA (left) and ProtoTTA (right) under noisy conditions. EATA suffers from hallucination, confidently activating irrelevant prototypes for an incorrect class (Predicted: 31, Ground Truth: 20) based on spurious background artifacts. In contrast, ProtoTTA accurately maps spatial features to the ground-truth class prototypes (Predicted: 20, Ground Truth: 20), effectively suppressing noise-induced hallucinations and restoring correct focus.

no downward trend in accuracy across later batches, indicating that our method does not suffer from *catastrophic forgetting* or error accumulation. The performance fluctuations align with the intrinsic difficulty of specific batches (mirrored by the unadapted baseline and EATA), confirming that our geometric filtering and reliability-weighted updates effectively prevent drift.

L Specification of the Adaptive Controller

Let $\mathcal{B} = \{x_i\}_{i=1}^{B}$ denote the current test batch, $m_i \in \{0, 1\}$ the reliability indicator defined below, and $\mathcal{R} = \{i : m_i = 1\}$. We use natural logarithms throughout, $H(\boldsymbol{p}_i) = -\sum_c p_{ic} \log p_{ic}$, and $H_{\mathrm{b}}(q) = -q \log q - (1 - q) \log(1 - q)$. Unless stated otherwise, $\epsilon = 10^{-8}$, $\epsilon_H = 10^{-6}$, and $\epsilon_W = 10^{-6}$. All confidence weights, controller statistics, gradient norms, routing decisions, and interpolation coefficients are detached before differentiating the final adaptation objective.

Backbone-specific prototype activations. Because the four architectures expose different native prototype responses, we map each response to $(0, 1)$ before computing prototype entropy or controller statistics.

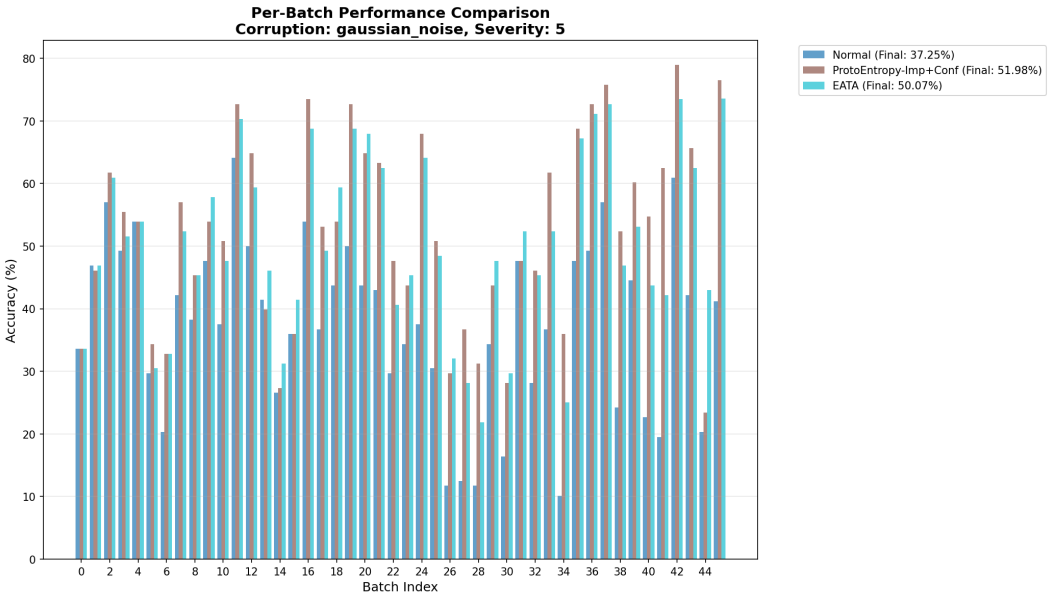


Figure 8: Per-Batch Performance Stability. We track classification accuracy across consecutive batches for Gaussian Noise (Severity 5). *ProtoTTA* (Brown bars) consistently outperforms the Unadapted baseline (Blue) and remains competitive with EATA (Cyan) throughout the sequence. The absence of performance degradation in later batches confirms that *ProtoTTA* avoids catastrophic forgetting, successfully maintaining stability by restricting updates to the reliable set.

For ProtoViT, let $s^{\cos}_{ipm}$ be the cosine similarity of sub-prototype $m \in \{1, \dots, 4\}$. We first use Top-2 mean aggregation,

$$u^{\mathrm{PV}}_{ip} = \mathrm{mean}\big(\mathrm{Top2}^{4}_{m=1}\, s^{\cos}_{ipm}\big)\,, \tag{28}$$

and map the result as

$$v^{\mathrm{PV}}_{ip} = \frac{\mathrm{clip}(u^{\mathrm{PV}}_{ip}, -1, 1) + 1}{2}, \qquad \bar{s}^{\mathrm{PV}}_{ip} = \mathrm{clip}\left(v^{\mathrm{PV}}_{ip}, \epsilon_H, 1 - \epsilon_H\right). \tag{29}$$

The loss uses $\bar{s}^{\mathrm{PV}}_{ip}$. Controller statistics use the same affine mapping with a lower numerical clamp of 10^{-8}.

For each ProtoPFormer branch $b \in \{\mathrm{local}, \mathrm{global}\}$, the native activation is the model's log-inverse-distance response

$$u^{b}_{ip} = \log\left(\frac{d^{b}_{ip} + 1}{d^{b}_{ip} + 10^{-4}}\right), \tag{30}$$

where d^{b}_{ip} is a squared Euclidean prototype distance. For the local branch, the maximum native activation over the exposed spatial/token locations is used; the global CLS branch already yields one response per prototype. The controller mapping is

$$v^{b}_{ip} = \sigma\big(\mathrm{ReLU}(u^{b}_{ip}) - 1\big)\,, \qquad \bar{s}^{b}_{ip} = \mathrm{clip}\left(v^{b}_{ip}, \epsilon_H, 1 - \epsilon_H\right), \tag{31}$$

where the clipped value is used in the prototype loss. The local and global prototype losses are combined with equal weight.

For ProtoPNet, the native response is obtained from the minimum spatial squared Euclidean distance,

$$d_{ip} = \min_{u,v} \mathrm{ReLU}\left(\|z_{iuv}\|^2_2 - 2z^{\top}_{iuv}p + \|p\|^2_2\right), \tag{32}$$

$$u^{\mathrm{PPN}}_{ip} = \frac{1}{9}\log\left(\frac{d_{ip} + 1}{d_{ip} + 10^{-4}}\right). \tag{33}$$

The divisor is exactly 9, and no sample- or batch-dependent normalization is applied. The unbounded value u_{ip}^{PPN} is used for geometric filtering. Controller statistics use $v_{ip}^{\mathrm{PPN}} = \mathrm{clip}(u_{ip}^{\mathrm{PPN}}, 0, 1)$, whereas the prototype loss uses

$$\bar{s}_{ip}^{\mathrm{PPN}} = \mathrm{clip}\left(u_{ip}^{\mathrm{PPN}}, \epsilon_H, 1-\epsilon_H\right). \tag{34}$$

For ProtoLens, let u_{ip}^{PL} be the cosine similarity to shared prototype p. The mapped activation is

$$v_{ip}^{\mathrm{PL}} = \sigma\big(5\ \mathrm{clip}(u_{ip}^{\mathrm{PL}}, -1, 1)\big), \qquad \bar{s}_{ip}^{\mathrm{PL}} = \mathrm{clip}\left(v_{ip}^{\mathrm{PL}}, \epsilon_H, 1-\epsilon_H\right). \tag{35}$$

Reliability filtering. All filters use the full prototype set rather than only prototypes associated with the pseudo-label. The reliability indicators used in the reported experiments are

$$m_i^{\mathrm{PV}} = \mathbb{1}\left[\max_p u_{ip}^{\mathrm{PV}} > 0.92\right], \tag{36}$$

$$m_i^{\mathrm{PF}} = \mathbb{1}\left[\frac{1}{2}\max_p u_{ip}^{\mathrm{local}} + \frac{1}{2}\max_p u_{ip}^{\mathrm{global}} > 0.55\right], \tag{37}$$

$$m_i^{\mathrm{PPN}} = \mathbb{1}\left[\max_p u_{ip}^{\mathrm{PPN}} > 0.8\right] \mathbb{1}\left[H(\boldsymbol{p}_i) < 0.4 \log 5\right], \tag{38}$$

$$m_i^{\mathrm{PL}} = \mathbb{1}\left[\mathrm{mean}\left(\mathrm{Top25}_{p=1}^{50}\, u_{ip}^{\mathrm{PL}}\right) \geq 0.1\right]. \tag{39}$$

Thus, an output-entropy constraint is used only for ProtoPNet. If $\mathcal{R}$ is empty, the batch does not produce an adaptation update.

Per-sample adaptation losses. Let $\hat{y}_i = \arg\max_c p_{ic}$ be the pseudo-label from the current pre-update forward pass. For ProtoViT, the prototype loss uses the predicted-class prototypes with normalized classifier-head weights,

$$w_{ip} = \frac{|W_{\hat{y}_i p}| \mathbb{1}[p \in \mathcal{P}_{\hat{y}_i}]}{\sum_{r \in \mathcal{P}_{\hat{y}_i}} |W_{\hat{y}_i r}| + \epsilon}, \qquad \ell_i^{\mathrm{proto}} = \sum_{p \in \mathcal{P}_{\hat{y}_i}} w_{ip} H_{\mathrm{b}}(\bar{s}_{ip}^{\mathrm{PV}}). \tag{40}$$

ProtoPFormer uses a uniform mean over the predicted-class prototypes within each branch,

$$\ell_i^{\mathrm{proto}} = \frac{1}{2}\frac{1}{|\mathcal{P}_{\hat{y}_i}^{\mathrm{local}}|} \sum_{p \in \mathcal{P}_{\hat{y}_i}^{\mathrm{local}}} H_{\mathrm{b}}(\bar{s}_{ip}^{\mathrm{local}}) + \frac{1}{2}\frac{1}{|\mathcal{P}_{\hat{y}_i}^{\mathrm{global}}|} \sum_{p \in \mathcal{P}_{\hat{y}_i}^{\mathrm{global}}} H_{\mathrm{b}}(\bar{s}_{ip}^{\mathrm{global}}). \tag{41}$$

ProtoPNet likewise uses the uniform mean over its ten predicted-class prototypes,

$$\ell_i^{\mathrm{proto}} = \frac{1}{|\mathcal{P}_{\hat{y}_i}|} \sum_{p \in \mathcal{P}_{\hat{y}_i}} H_{\mathrm{b}}(\bar{s}_{ip}^{\mathrm{PPN}}). \tag{42}$$

ProtoLens uses all $P = 50$ shared prototypes. Its classifier-derived importance and soft target are

$$a_{ip} = \frac{|W_{\hat{y}_i p}|}{\max_r |W_{\hat{y}_i r}| + \epsilon_W}, \qquad t_{ip} = \sigma(2W_{\hat{y}_i p}), \tag{43}$$

and its prototype loss is

$$\ell_i^{\mathrm{proto}} = \frac{1}{P}\sum_{p=1}^{P} a_{ip}\left[-t_{ip} \log \bar{s}_{ip}^{\mathrm{PL}} - (1-t_{ip}) \log(1-\bar{s}_{ip}^{\mathrm{PL}})\right]. \tag{44}$$

For every architecture, the output component is $\ell_i^{\mathrm{out}} = H(\boldsymbol{p}_i)$. ProtoViT and ProtoPFormer use the detached confidence weight $c_i = \max_c p_{ic}$ for both components. Confidence weighting is disabled for ProtoPNet and ProtoLens, for which $c_i = 1$.

Activation-margin coefficient. For a set of eligible mapped activations $\mathcal{V}_i$, define

$$\Lambda_{\mathrm{act}}(\mathcal{V}_i) = \mathrm{clip}\left(\frac{\mathrm{mean}_{v\in \mathrm{Top3}(\mathcal{V}_i)}\,|v-\frac{1}{2}|}{\delta_0}, 0, 1\right), \qquad \delta_0 = 0.25. \tag{45}$$

We use $k = 3$ and reference margin $\delta_0 = 0.25$ for all architectures. If fewer than three eligible activations are available, the mean is taken over all available activations. ProtoViT and ProtoPNet set $\mathcal{V}_i$ to the mapped activations of prototypes in $\mathcal{P}_{\hat{y}_i}$. ProtoPFormer computes the coefficient separately for the local and global predicted-class prototypes and uses

$$\lambda_i^{\mathrm{act}} = \frac{1}{2}\Lambda_{\mathrm{act}}\left(\mathcal{V}_i^{\mathrm{local}}\right) + \frac{1}{2}\Lambda_{\mathrm{act}}\left(\mathcal{V}_i^{\mathrm{global}}\right). \tag{46}$$

For ProtoLens, $\mathcal{V}_i = \{v_{ip}^{\mathrm{PL}} : W_{\hat{y}_i p} > 0\}$; if this set is empty, the Top-3 shared-prototype activations over all prototypes are used. All other architectures use $\lambda_i^{\mathrm{act}} = \Lambda_{\mathrm{act}}(\mathcal{V}_i)$.

Relative-evidence coefficient. For ProtoViT and ProtoPNet, the evidence for class c is

$$S_{i,c} = \mathrm{mean}\left(\mathrm{Top3}_{p\in\mathcal{P}_c}\, v_{ip}\right), \tag{47}$$

using the corresponding mapped controller activations. For ProtoPFormer, the mapped local and global activations belonging to class c are concatenated before the Top-3 mean is taken. For ProtoLens, let

$$\mathcal{C}_c = \{p : W_{cp} > 0\}. \tag{48}$$

If $\mathcal{C}_c$ is empty, it is replaced by the three prototypes having the largest values of W_{cp}, and $S_{i,c}$ is the Top-3 mean over $p \in \mathcal{C}_c$.

Let $p_i^{(1)}$ and $p_i^{(2)}$ be the largest and second-largest output probabilities. We define

$$r_i^{\mathrm{proto}} = \mathrm{clip}\left(\frac{[S_{i,\hat{y}_i} - \max_{c\neq\hat{y}_i} S_{i,c}]_+}{|S_{i,\hat{y}_i}| + \epsilon}, 0, 1\right), \tag{49}$$

$$r_i^{\mathrm{out}} = \mathrm{clip}\left(\frac{p_i^{(1)} - p_i^{(2)}}{p_i^{(1)} + \epsilon}, 0, 1\right), \tag{50}$$

$$\lambda_i^{\mathrm{rel}} = \frac{r_i^{\mathrm{proto}}}{r_i^{\mathrm{proto}} + r_i^{\mathrm{out}} + \epsilon}. \tag{51}$$

Gradient consistency and batch-level routing. Let $\mathcal{I} = (i : m_i = 1)$ preserve the order of samples in the current batch, and form the interleaved subsets $\mathcal{I}_1 = \mathcal{I}[1, 3, 5, \ldots]$ and $\mathcal{I}_2 = \mathcal{I}[2, 4, 6, \ldots]$. For $j \in \{\mathrm{proto}, \mathrm{out}\}$ and $h \in \{1, 2\}$, define

$$\boldsymbol{g}_{j,h} = \nabla_\theta \frac{1}{|\mathcal{I}_h|} \sum_{i\in\mathcal{I}_h} c_i \ell_i^j. \tag{52}$$

Missing parameter gradients are omitted (equivalently, replaced by zeros) when forming the concatenated gradient vector. The consistency score is

$$q_j = \left[\frac{\boldsymbol{g}_{j,1}^\top \boldsymbol{g}_{j,2}}{\|\boldsymbol{g}_{j,1}\|_2 \|\boldsymbol{g}_{j,2}\|_2 + \epsilon_q}\right]_+ \frac{2\min(\|\boldsymbol{g}_{j,1}\|_2, \|\boldsymbol{g}_{j,2}\|_2)}{\|\boldsymbol{g}_{j,1}\|_2 + \|\boldsymbol{g}_{j,2}\|_2 + \epsilon_q}. \tag{53}$$

We set $q_j = 0$ if fewer than four reliable samples are available or if either half-gradient is zero. We use $\epsilon_q = 10^{-12}$ for ProtoViT, ProtoPFormer, and ProtoLens, and $\epsilon_q = 10^{-8}$ for ProtoPNet.

The consistency gate is

$$G^{\mathrm{cons}} = \mathbb{1}\left[q_{\mathrm{proto}} \geq 2q_{\mathrm{out}}^\dagger\right] \mathbb{1}\left[q_{\mathrm{proto}} \geq 0.25\right], \tag{54}$$

where $q_{\mathrm{out}}^\dagger = \max(q_{\mathrm{out}}, 10^{-8})$ for ProtoViT, ProtoPFormer, and ProtoLens, and $q_{\mathrm{out}}^\dagger = q_{\mathrm{out}}$ for ProtoPNet. Define the reliable-sample fraction and mean output margin as

$$\rho = \frac{|\mathcal{R}|}{B}, \qquad \bar{r}^{\mathrm{out}} = \frac{1}{|\mathcal{R}|}\sum_{i\in\mathcal{R}} r_i^{\mathrm{out}}. \tag{55}$$

The coverage guard and final route are

$$G^{\mathrm{cov}} = \mathbb{1}\left[\bar{r}^{\mathrm{out}} \geq 0.90\right] \mathbb{1}\left[\rho \leq 0.40\right], \tag{56}$$

$$G = G^{\mathrm{cons}} \vee G^{\mathrm{cov}}, \tag{57}$$

$$\lambda_i = G\lambda_i^{\mathrm{act}} + (1-G)\lambda_i^{\mathrm{rel}}. \tag{58}$$

The route G is shared across the reliable samples in a batch, while the selected coefficient remains sample-specific. The coefficients are allowed to attain both endpoints of $[0, 1]$, and no temporal smoothing is applied.

Gradient normalization and final objective. For $j \in \{\mathrm{proto}, \mathrm{out}\}$, the raw reliable-set component and its global parameter-gradient norm are

$$L_j = \frac{1}{|\mathcal{R}| + \epsilon} \sum_{i \in \mathcal{R}} c_i \ell_i^j, \tag{59}$$

$$g_j = \left(\sum_{\theta_t \in \theta} \|\nabla_{\theta_t} L_j\|_2^2 \right)^{1/2}. \tag{60}$$

Thus, g_j is the norm of the conceptual concatenation of the gradients of all enabled adaptation parameters. Missing gradients are omitted. The detached norms are used in the final objective

$$\mathcal{L}_\lambda = \frac{1}{|\mathcal{R}| + \epsilon} \sum_{i \in \mathcal{R}} c_i \left[\lambda_i \frac{\ell_i^{\mathrm{proto}}}{g_{\mathrm{proto}} + \epsilon} + (1-\lambda_i) \frac{\ell_i^{\mathrm{out}}}{g_{\mathrm{out}} + \epsilon} \right]. \tag{61}$$

If a component gradient is zero, its denominator reduces to ϵ; a constant component nevertheless contributes no parameter update.

Adapted parameters and optimization. For ProtoViT and ProtoPFormer, θ contains the LayerNorm affine parameters and attention-module biases. ProtoLens adapts its LayerNorm affine parameters and attention-projection biases. ProtoPNet adapts the affine parameters of all 16 VGG19-BN normalization layers and the weights of the two 1×1 add-on convolutions; its prototype vectors, classifier head, and remaining convolutional weights are frozen. ProtoPNet does not accumulate running normalization statistics: its BatchNorm buffers are disabled and current-batch statistics are used during evaluation. ProtoViT and ProtoLens are adapted in evaluation mode, whereas ProtoPFormer is adapted in training mode.

All experiments use Adam (Kingma & Ba, 2017) with $(\beta_1, \beta_2) = (0.9, 0.999)$, zero weight decay, and one update per accepted batch. ProtoViT and ProtoPFormer use learning rate 10^{-3} and batch size 128; ProtoPNet uses learning rate 10^{-3} and batch size 64; ProtoLens uses learning rate 5×10^{-6} and batch size 128. Adaptation is continual within each corruption setting. The source checkpoint is restored between visual corruption types and between ProtoLens corruption–severity settings.