MITIGATING CACHE NOISE IN TEST-TIME ADAPTATION FOR LARGE VISION-LANGUAGE MODELS

Haotian Zhai[1,2,*], **Xinyu Chen**[2,*], **Can Zhang**[2,*], **Tianming Sha**[3], **Ruirui Li**[2,†]
[1]University of Minnesota Twin Cities, [2]Beijing University of Chemical Technology, [3]Sun Yat-sen University
zihan9198@gmail.com, centurychen1103@gmail.com, alexlessend@gmail.com, shatm@mail2.sysu.edu.cn, liruirui@mail.buct.edu.cn

ABSTRACT

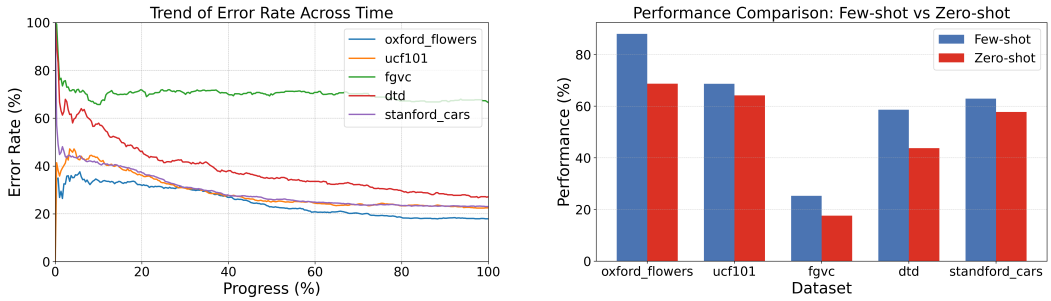
Test-time adaptation (TTA) of visual language models has recently attracted significant attention as a solution to the performance degradation caused by distribution shifts in downstream tasks. However, existing cache-based TTA methods have certain limitations. They mainly rely on the accuracy of cached feature labels, and the presence of noisy pseudo-labels can cause these features to deviate from their true distribution. This makes cache retrieval methods based on similarity matching highly sensitive to outliers or extreme samples. Moreover, current methods lack effective mechanisms to model class distributions, which limits their ability to fully exploit the potential of cached information. To address these challenges, we introduce a comprehensive and reliable caching mechanism and propose a novel zero-shot TTA method called "Cache, Residual, Gaussian" (CRG). This method not only employs learnable residual parameters to better align positive and negative visual prototypes with text prototypes, thereby optimizing the quality of cached features, but also incorporates Gaussian Discriminant Analysis (GDA) to dynamically model intra-class feature distributions, further mitigating the impact of noisy features. Experimental results on 13 benchmarks demonstrate that CRG outperforms state-of-the-art TTA methods, showcasing exceptional robustness and adaptability.

1 INTRODUCTION

In recent years, vision-language models pre-trained on large-scale datasets, such as CLIP [1], have demonstrated remarkable zero-shot capabilities in downstream tasks. However, the direct application of un-tuned CLIP in practical scenarios often yields suboptimal performance due to distributional shifts between training and downstream data. Moreover, obtaining even a small number of high-quality annotated samples can be impractical and costly in certain situations. Consequently, effectively adapting vision-language models in zero-shot settings has emerged as a significant research focus and technical challenge. A series of Test-Time Adaptation (TTA) [2, 3, 4, 5] methods have been introduced to dynamically address distribution shifts during the testing phase of vision-language models. Recently, a cache-based test-time adaptation method called TDA [5] has been proposed. TDA maintains a lightweight cache during testing to store representative test samples, guiding the classification of subsequent samples.

We conducted an extensive review of the literature and further compared the performance of cache-based training-free methods through experiments. Tip-adapter[6] is a cache-based few-shot method, whereas TDA[5] is a zero-shot method. Our observations revealed a significant performance gap between zero-shot and few-shot settings. Fig. 1b illustrates the performance of these methods under zero-shot and 8-shot conditions, demonstrating that the performance in the few-shot setting is significantly superior to that in the zero-shot setting. In these two configurations, the main difference lies in the content stored within the cache: the zero-shot method's cache is populated using pseudo-labels. To further investigate, we analyzed the annotation accuracy of cached samples in the zero-shot setting, as shown in Fig. 1a. The error rate starts relatively high and decreases over the

* These authors contributed equally to this work.
† Corresponding author.



(a) Error Rate Plot

(b) Few-shot vs Zero-shot Performance

Figure 1: Figure (a) shows the change in error rates for the positive cache during downstream testing with the TDA[5] method, while Figure (b) compares the performance gap between similarity-based cache models in zero-shot(TDA[5]) and few-shot (Tip-adapter[6]) settings, all using RN50 as the backbone.

course of training but consistently remains above 20%. We posit that these noisy labels contribute to an imbalance in the features stored in the cache and cause class prototypes to deviate from the true distribution. Unfortunately, previous research has not effectively addressed these noisy labels nor fully explored the distribution of features stored in the cache.

To address these limitations, we propose the "Cache, Residual, Gaussian" (CRG) zero-shot test-time adaptation method, which enhances cache reliability via a comprehensive mechanism. CRG maintains two caches for positive and negative image features, plus a cache for text prototypes. Image prototypes are derived by averaging features within each class, and learnable residual vectors are used to align text and image prototypes by minimizing prediction entropy and maximizing inter-prototype distance. However, noisy samples may persist in the cache. To mitigate their impact, CRG employs Gaussian Discriminant Analysis (GDA) and Bayes' theorem to model class distributions and compute posterior probabilities. This distribution-based decision-making reduces noise interference, while negative class prototypes further curb overconfidence caused by corrupted features. Through the synergy of GDA and negative prototypes, CRG achieves robust, accurate performance even under noisy conditions.

Our contributions can be summarized as follows:

- We have analyzed the factors limiting Test-Time Adaptation (TTA) performance and identified noisy labels as a key factor in the performance gap between zero-shot and few-shot settings.
- We propose a novel TTA framework that effectively enhances robustness and generalization under noisy conditions by leveraging Gaussian Discriminant Analysis (GDA) and negative prototype learning.
- During the TTA process, we simultaneously minimize prediction entropy and maximize inter-prototype distances, achieving effective prototype calibration based on learnable residuals, thereby improving overall adaptation performance.

2 Related Works

2.1 Vision-Language Models

Recent years have witnessed rapid progress in foundational models, particularly vision-language models, due to their impressive generalization capabilities. Specifically, CLIP aligns vision and text embeddings through contrastive learning on large-scale, diverse image-text datasets. However, due to the distribution shift between training and downstream data, many researchers have explored fine-tuning CLIP with a small amount of labeled data to address distribution shift and improve adaptation. Existing CLIP fine-tuning approaches are mainly categorized into two types[7]: prompt-based methods (e.g., CoOp[8] and Maple[9]) and adapter-based methods (e.g., Tip-Adapter[6] and

CLIP-Adapter[10]). Nonetheless, these few-shot fine-tuning methods are limited to the distribution of a small number of labeled samples, restricting their generalization to broader distributions. In this study, we focus on test-time adaptation under zero-shot settings, as acquiring labeled samples in real-world deployments is often impractical.

2.2 Test-Time Adaptation for Vision-Language Models

Recently, test-time adaptation of large-scale vision-language models has emerged as a research hotspot. TPT[2] (Test-Time Prompt Tuning) integrates consistency regularization with prompt tuning to enforce consistency between augmented views of test samples. DiffTPT[3] improves upon TPT with stronger augmentations but incurs high computational costs. PromptAlign[4] aligns visual and textual features using statistical characteristics of a proxy dataset. TDA[5] introduces a lightweight cache to store low-entropy historical test features to guide classification. However, these methods often require gradient backpropagation or remain vulnerable to noise in cached features.To address these challenges, we design CRG. CRG aligns prototypes by learning residuals without requiring backpropagation through the encoder. By integrating Gaussian Discriminant Analysis[11] (GDA) and negative learning, it achieves greater robustness under noisy label conditions.

3 Method

In this section, we first provide a brief overview of the foundational concepts of CLIP[1]. We then present our method by following the sequence of Cache, Residual, and Gaussian, outlining each component in detail to highlight how they contribute to the overall framework.

3.1 Preliminaries

Contrastive Language-Image Pre-training (CLIP). CLIP[1] utilizes two pre-trained parallel encoders: a visual encoder $E_V(\cdot)$ and a textual encoder $E_T(\cdot)$, which embed images and text descriptions into a shared embedding space $\mathbb{R}^d$. For a C-class classification task, CLIP performs zero-shot predictions by computing the similarities between the extracted image feature and the C candidate text features, formulated as:

$$P_{\text{CLIP}}(y = c \mid X_{\text{test}}) = \frac{\exp\left(\text{sim}\left(E_T(T_c), E_V(X_{\text{test}})\right)/\tau\right)}{\sum_{j=1}^{C} \exp\left(\text{sim}\left(E_T(T_j), E_V(X_{\text{test}})\right)/\tau\right)}, \tag{1}$$

where $X_{\text{test}} \in D_{\text{test}}$ denotes the input test image, and T_c represents the class-specific text description for class y_c. The pairwise similarity $\text{sim}(\cdot, \cdot)$ is calculated using cosine similarity, and τ is the temperature parameter in the softmax function.

3.2 Cache

Inspired by the method of TDA[5], we adopt a priority queue strategy to store image features for each class. Specifically, for each test image X_{test}, after making a prediction, we assign it to the corresponding class queue based on its predicted pseudo-label $\hat{y}$. Meanwhile, we compute the entropy value h_{test} of the test image and store the image feature together with its entropy as a pair $(E_V(X_{\text{test}}), h_{\text{test}})$ in the queue. For each class c, the size of its priority queue is $\mathcal{M}$, and each queue is initialized with a text feature t_c, the reasoning for which will be detailed in the Gaussian section.

When the queue is full, we compare the entropy value h_{test} of the new test image with the highest entropy value in the queue: if the new image has higher entropy, it will be discarded; if the entropy is lower, the image with the highest entropy in the queue will be replaced by the new image. If the queue is not yet full, the new image is inserted directly.

We also construct a text-side cache T_{cache} to store class-specific text features derived from text descriptions. The cache has the shape $T_{\text{cache}} \in \mathbb{R}^{C \times d}$, where

$$T_{\text{cache}} = [t_1, t_2, \ldots, t_C],$$

and each t_c represents the text feature for class c.

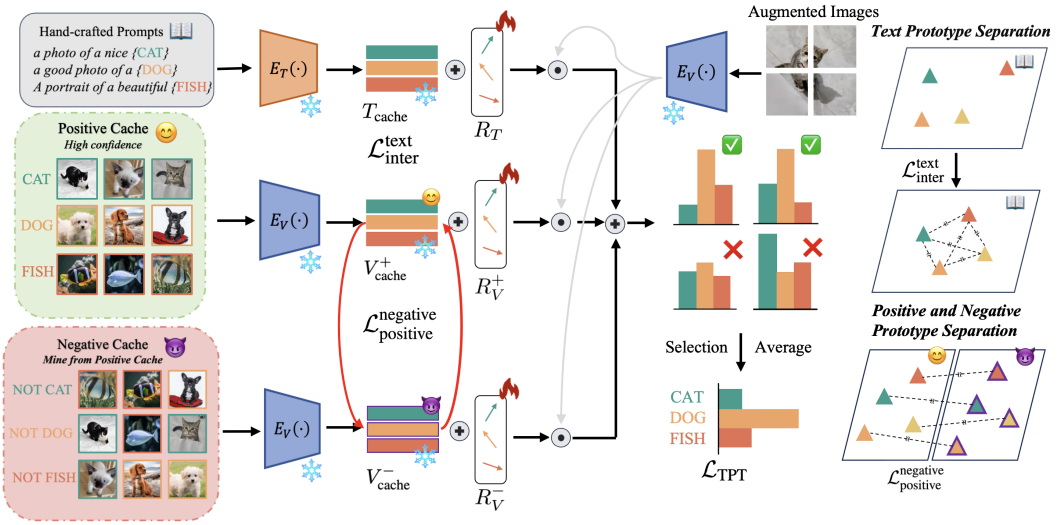


Figure 2: **Overview of the CRG Method.** We introduce caches (both positive and negative) in the text and visual modalities and employ learnable residual vectors to flexibly calibrate multi-modal features. During inference, positive prototypes, negative prototypes, and text prototypes are used for similarity matching, while GDA models the feature distribution to mitigate noise interference in predictions. By simultaneously minimizing prediction entropy and maximizing inter-prototype distances, CRG achieves multi-modal alignment with enhanced robustness and generalization.

To incorporate distribution information from historical samples, T_{cache} is continuously updated during inference. Specifically, let T^*_{cache} be the text cache optimized for each test sample in Equation 2. To retain historical information, we update the text-side over time like DPE[12]

$$T_{\text{cache}} \leftarrow \frac{(k-1)T_{\text{cache}} + T^*_{\text{cache}}}{\|(k-1)T_{\text{cache}} + T^*_{\text{cache}}\|}, \quad k \leftarrow k+1$$

To ensure stable online updates, we set an entropy threshold τ_t to filter out low-confidence samples (for which $H(P(X_{\text{test}})) < \tau_t$) from updating the online prototypes, and maintain a counter k for tracking confident samples.

3.3 Residual

We observe that residual learning holds significant potential [13] for improving model performance. More importantly, we introduce learnable residuals on both the text and vision sides, enabling feature space calibration for each test sample during inference. During testing, we learn three types of residuals: text cache residuals and vision cache residuals, with the latter further divided into positive and negative cache residuals.

Text Cache Residuals. We introduce learnable residual parameters R_T to dynamically adjust the text cache during testing. Specifically, the text cache is updated as follows:

$$T_{\text{cache}} = \text{Normalize}(T_{\text{cache}} + R_T) \tag{2}$$

Here, $R_T \in \mathbb{R}^{C \times d}$ is a learnable parameter initialized to zero.

Positive Cache Residuals. For each class c, we compute the average of the features stored in the visual cache to obtain a cache prototype representing each class, denoted as $V^+_{\text{cache}} = [v^+_1, v^+_2, \ldots, v^+_C] \in \mathbb{R}^{C \times d}$. Subsequently, we introduce a set of visual residuals R^+_V to further refine these prototypes. Specifically, the updated visual prototype is computed as:

$$V^+_{\text{cache}} = \text{Normalize}(V^+_{\text{cache}} + R^+_V) \tag{3}$$

where $R_V^+ \in \mathbb{R}^{C\times d}$ is the learnable residual parameter, initialized to zero.

Negative Cache Residuals. Unlike TDA[5], which treats high-entropy samples as negative samples, our approach leverages the principle that if an image is classified as a dog, it cannot simultaneously be a cat. Therefore, instead of constructing a separate negative sample cache, we mine the negative prototype directly from the existing cache.

To construct the negative cache prototypes $V_{\text{cache}}^- = [v_1^-, v_2^-, \ldots, v_n^-] \in \mathbb{R}^{C\times d}$, for each class c, we compute the average of the prototypes from all other classes, effectively excluding class c. Formally, the negative prototype is defined as:

$$v_c^- = \frac{1}{C-1}\sum_{\substack{j=1\\ j\neq c}}^{C} v_j^+ \tag{4}$$

To further refine these negative prototypes, we introduce a set of learnable residual parameters $R_V^- \in \mathbb{R}^{C\times d}$. The updated negative prototypes are computed as:

$$V_{\text{cache}}^- = \text{Normalize}\left(V_{\text{cache}}^- + R_V^-\right) \tag{5}$$

where V_{cache}^- is the set of the negtive protptypes and $R_V^- \in \mathbb{R}^{C\times d}$ is the learnable residual parameter, initialized to zero.

Test-Time Logit Inference and Loss Function.

The learning diagram of CRG can be illustrated by Fig.2. Given the complexity of Gaussian Discriminant Analysis and the risk of instability in high-dimensional spaces, we continue to use the similarity matching method to adjust class prototypes. Using the text cache, positive cache prototypes, and negative cache prototypes, the prediction for the input sample X is as follows:

$$P(y=c \mid X) = \frac{\exp\left((f_v^\top t_c + \mathcal{A}(f_v^\top v_c^+) + \mathcal{B}(f_v^\top v_c^-))/\tau\right)}{\sum_{j=1}^{C}\exp\left((f_v^\top t_j + \mathcal{A}(f_v^\top v_j^+) + \mathcal{B}(f_v^\top v_j^-))/\tau\right)} \tag{6}$$

Here, $\top$ denotes the matrix transpose, $\mathcal{A}(x) = \lambda_1 \exp(-\beta(1-x))$ is the logits for positive sample inference, where λ_1 is the balance parameter for positive samples and β is the sharpness ratio; $\mathcal{B}(x) = \lambda_2 \exp(\beta(1-x))$ represents the logits for negative sample inference, where λ_2 is the balance parameter for negative samples.

Similar to the loss function used in TPT[2], we optimize these three residuals to promote consistent predictions across N different augmented views of the given test image X_{test} using the unsupervised entropy minimization objective.

$$\mathcal{L}_{\text{TPT}} = H\left(\frac{1}{\rho N}\sum_{i=1}^{N}\mathbb{I}\left(H(P(y \mid \tilde{X}_i)) < \theta\right) P(y \mid \tilde{X}_i)\right), \tag{7}$$

where $P(y \mid \tilde{X}_i)$ is the predicted probability distribution for the augmented view $\tilde{X}_i$, $H(\cdot)$ is the entropy function, θ is an entropy threshold, ρ represents the filtered ratio, and $\mathbb{I}(\cdot)$ is the indicator function ensuring that only augmented views with low entropy are used for training.

However, in traditional TPT[2], there are no cached features. When we work with a cache that has labeled samples and test samples without labels, this essentially becomes an unsupervised domain adaptation problem. We believe that during test time, we should aim to separate prototypes of different categories as much as possible. At the same time, we also require that positive cache prototypes and negative cache prototypes be well separated. Based on this, we define the following loss function:

$$\mathcal{L}_{\text{inter}}^{\text{text}} = \sum_{m\neq n}\exp\left(-\gamma\|t_m - t_n\|_2^2\right) \tag{8}$$

$$\mathcal{L}_{\text{positive}}^{\text{negative}} = \sum_{c=1}^{C} \frac{\langle v_c^+, v_c^- \rangle}{\|v_c^+\| \|v_c^-\|}. \tag{9}$$

Here, Equation 8 is used to separate the text prototype, while Equation 9 is used to separate the positive cache prototype and the negative cache prototype, where t_m is the text features of the m-th class, v_c^+ denotes the positive cache prototype, and v_c^- denotes the negative cache prototype.

Overall, our final optimization objective is:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{TPT}} + \xi_1 \mathcal{L}_{\text{inter}}^{\text{text}} + \xi_2 \mathcal{L}_{\text{positive}}^{\text{negative}}. \tag{10}$$

Through these approaches, we can achieve better alignment of the three different prototype features.

3.4 GAUSSIAN

Theoretical Justification. Recent studies[14] have theoretically demonstrated that features follow a Gaussian distribution when the network is trained with the Softmax function. Since CLIP uses the Softmax function during training, this provides a theoretical justification for applying Gaussian Discriminant Analysis[11]. In previous calibration processes, learning a residual essentially corresponds to applying a simple linear transformation to the features. If the original features follow a Gaussian distribution, the calibrated features will also retain the Gaussian distribution property.

Gaussian Discriminant Analysis. We first introduce the assumption of GDA[11], where the features x under class i follow a Gaussian distribution. Specifically, the class-conditional distribution can be expressed as:

$$p(x \mid y = i) \sim \mathcal{N}(\mu_i, \Sigma),$$

where μ_i is the mean vector for class i, and Σ is the shared covariance matrix across all classes.

Based on this assumption, we use Bayes' theorem to compute the posterior probability $p(y = i \mid x)$, given by:

$$p(y = i \mid x) = \frac{p(x \mid y = i) p(y = i)}{\sum_{j=1}^{C} p(x \mid y = j) p(y = j)}.$$

Substituting the Gaussian form of $p(x \mid y = i) \sim \mathcal{N}(\mu_i, \Sigma)$, into the above equation, the posterior probability can be further written as follows. The detailed derivation process is provided in the appendix. :

$$p(y = i \mid x) = \frac{\exp\left(\mu_i^T \Sigma^{-1} x - \frac{1}{2} \mu_i^T \Sigma^{-1} \mu_i + \log p_i\right)}{\sum_{j=1}^{C} \exp\left(\mu_j^T \Sigma^{-1} x - \frac{1}{2} \mu_j^T \Sigma^{-1} \mu_j + \log p_j\right)},$$

where $p_i = p(y = i) = \frac{1}{C}$ for $i = 1, 2, \ldots, C$, assuming a uniform prior distribution across all classes.

Through normalization of the posterior probability, the classifier can be equivalently represented as a linear classifier, where the weight w_i and bias b_i are defined as:

$$w_i = \Sigma^{-1} \mu_i, \quad b_i = \log p_i - \frac{1}{2} \mu_i^T \Sigma^{-1} \mu_i.$$

Finally, the decision function for the classifier is given by:

$$h_i(x) = w_i^T x + b_i. \tag{11}$$

In the actual inference process, we integrate the learned residuals into the cached features of each category to obtain well-aligned positive cache features. We then compute the mean for each class and the overall covariance to construct a Gaussian discriminant classifier, which replaces the original positive sample inference method. Additionally, to prevent some category queues from being empty at initialization, we insert a pair consisting of a textual vector and a high-entropy value into each category queue.

Finally, for the final inference, we replace $\mathcal{A}(f_v^\top v_c^+)$ with $\lambda_1 h_c(f_v)$. Following the format of Equations (6) and (11), this ultimately leads to Equation (12).

$$P(y=c \mid X) = \frac{\exp\left((f_v^\top t_c + \lambda_1 h_c(f_v) + \mathcal{B}(f_v^\top v_c^-))/\tau\right)}{\sum_{j=1}^{C} \exp\left((f_v^\top t_j + \lambda_1 h_i(f_v) + \mathcal{B}(f_v^\top v_j^-))/\tau\right)} \tag{12}$$

4 EXPERIMENT

4.1 DATASETS.

We follow previous work[5] to evaluate our method on two benchmarking scenarios: cross-dataset generalization and robustness to natural distribution shifts. (1) For cross-dataset generalization tasks, we conduct comprehensive assessments across 10 diverse recognition datasets, including FGVC Aircraft[15], Caltech101[16], Stanford Cars[17], DTD[18], EuroSAT[19], Flowers102[20], Food101[21], Oxford Pets[22], SUN397[23], and UCF101[24]. These datasets provide a comprehensive benchmark for evaluating the adaptability and generalization ability of methods across different datasets. (2) For evaluating robustness to natural distribution shifts, we assess the performance of our method using the ImageNet[25] dataset alongside its variant out-of-distribution datasets, including ImageNet-A[26], ImageNet-V2[27]. This evaluation measures our method's robustness in the presence of different distribution shifts.It should be noted that for datasets with excessively large test sets, we only extract the 2000 samples from the test set, similar to DiffTPT[3].

4.2 IMPLEMENTATION DETAILS

We adopt ResNet-50 and ViT-B/16 as the visual encoders for CLIP. We use hand-crafted prompt, which are detailed in the appendix. Following the approach of TPT [2], we generate 63 augmented views for each test image. For the learning of three residual parameters, we utilize the AdamW optimizer with a learning rate of 0.0005, completing the optimization in a single iteration. In default, the hyperparameters are set as follows: ξ_1 and ξ_2 are set to 1 and 10, λ_1 and λ_2 are set to 7 and 0.3, τ_t is set to 0.1, ρ is set to 0.1, β is set to 5.0, and the queue size $\mathcal{M}$ is 12, much larger than TDA's, since GDA is more robust to noisy pseudo-labels and additional samples enhance class distribution modeling. All experiments are conducted on a single NVIDIA GTX 4090 GPU with 24GB of memory.

Table 1: Performance comparisons on robustness to natural distribution shifts. The best results are highlighted in **bold**. The asterisk (*) indicates that our method is evaluated on a subset of the test set.

Method	Publication	ImageNet	-A	-V2	Average
CLIP-ResNet-50	ICML 2021	58.16	21.83	56.15	44.18
CoOp [8]	IJCV 2022	63.33	23.06	**56.60**	46.66
TPT [2]	NIPS 2022	60.74	26.67	59.11	48.84
DiffTPT [3]	ICCV 2023	60.80	**31.06**	55.80	49.22
TDA [5]	CVPR 2024	61.35	30.29	55.54	49.06
Ours	N/A	**65.26***	28.34	56.07*	**49.89**
CLIP-ViT-B/16	ICML 2021	66.73	47.87	60.86	58.49
CoOp [8]	IJCV 2022	71.51	49.71	64.20	61.81
TPT [2]	NIPS 2022	68.98	54.77	63.45	62.40
DiffTPT [3]	ICCV 2023	70.30	55.68	**65.10**	63.69
TDA [5]	CVPR 2024	69.51	60.11	64.67	64.76
Ours	N/A	**75.01***	**63.67**	64.66*	**67.78**

4.3 COMPARISONS WITH STATE-OF-THE-ART

Robustness to Natural Distribution Shifts. Our approach consistently demonstrates strong generalization performance on downstream tasks with natural distribution shifts. As shown in Table 1, our

Table 2: Performance comparisons on cross-datesets generalization. The best results are highlighted in **bold**. The asterisk (*) indicates that our method is evaluated on a subset of the test set.

Method	Aircraft	Caltech	Cars	DTD	EuroSAT	Flower	Food101	Pets	SUN397	UCF101	Average
CLIP-ResNet-50	15.66	85.88	55.70	40.37	23.69	61.75	73.97	83.57	58.80	58.84	55.82
CoOp [8]	15.12	86.53	55.32	37.29	26.20	61.55	75.59	**87.00**	58.15	59.05	56.18
TPT [2]	17.58	87.02	58.46	40.84	28.33	62.69	74.88	84.49	61.46	60.82	57.66
DiffTPT [3]	17.60	86.89	**60.71**	40.72	41.04	63.53	**79.21**	83.40	62.72	62.67	59.85
TDA [5]	17.61	89.70	57.78	43.74	42.11	68.74	77.75	86.18	62.53	**64.18**	61.03
Ours	**18.09**	**90.12**	57.92	**51.89**	**46.80**	**71.09**	75.76*	85.91	**63.11***	63.58	**62.42**
CLIP-ViT-B/16	23.67	93.35	65.48	44.27	42.01	67.44	83.65	88.25	62.59	65.13	63.58
CoOp [8]	18.47	93.70	64.51	41.92	46.39	68.71	85.30	89.14	64.15	66.55	63.88
TPT [2]	24.78	94.16	66.87	47.75	42.44	68.98	84.67	87.79	65.50	68.04	65.10
DiffTPT [3]	25.60	92.49	67.01	47.00	43.13	70.10	**87.23**	88.22	65.74	62.67	65.47
TDA [5]	23.91	**94.24**	**67.28**	47.40	58.00	71.42	86.14	88.63	67.62	**70.66**	67.53
Ours	**26.58**	93.57	66.89	**53.78**	**59.81**	**75.94**	84.81*	**91.20**	**68.36***	70.31	**69.12**

method achieved the highest average results across datasets with significant distribution differences, each evaluated using different backbone networks. Specifically, we achieved the best performance on ImageNet, with improvements of 3.91 and 5.5 percentage points. It is noteworthy that the CoOp results were obtained using few-shot learning. Our method not only surpasses TDA but in certain cases also outperforms few-shot methods, showcasing the superiority of our approach. Although there are instances where our performance is not as strong as DiffTPT, we speculate this may be due to the lack of extensive data augmentation. Our method can be integrated with data augmentation techniques, but this was not the primary focus of our study.

Cross-Datasets Generalization. In Table 2, we further evaluate the generalization capability of our proposed method against other state-of-the-art methods across 10 fine-grained recognition datasets. Due to the substantial distribution differences among these datasets, performance can vary considerably. Despite this, our method achieves average improvements of 1.39 and 1.59 over the current best-performing methods on two distinct backbones, surpassing them on 6 out of the 10 datasets.

Notably, we observe a marked performance gain on the DTD dataset, likely because its strong emphasis on textures makes the underlying data distribution crucial for texture description. Conversely, our results on Food101 lag behind the current SOTA, which may be attributed to the high inter-class similarity that challenges GDA in accurately modeling class distributions. Overall, these findings underscore the remarkable robustness and adaptability of our approach when transferring to new domains at test time—a critical factor for real-world applications.

4.4 Discussion

Ablation Analysis. To further analyze the effectiveness of our method, we conducted an ablation study to examine the impact of different components in Table 3. Simply adding the negative cache and introducing learnable residual parameters may lead to ineffective parameter learning if there is no robust representation or adequate constraint guidance, which in turn can result in misalignment between different modalities. As observed, the performance change in this case is nearly zero. By introducing GDA and imposing constraints, we not only achieve more accurate inference but also facilitate effective prototype alignment. The ablation results demonstrate that these modules work together to fully leverage the trustworthy cache mechanism, ultimately improving the overall accuracy of the model.

Table 3: Ablation studies for different variants of our method.

T_{cache}	V^{+}_{cache}	V^{-}_{cache}	GDA	$\mathcal{L}^{\text{text}}_{\text{inter}}$	$\mathcal{L}^{\text{negative}}_{\text{positive}}$	Flowers	ImageNet
✓	✓	✗	✗	✗	✗	73.35	73.27
✓	✓	✗	✗	✓	✗	73.57	73.56
✓	✓	✓	✗	✗	✗	73.35	73.37
✓	✓	✓	✗	✗	✓	73.57	73.66
✓	✓	✗	✓	✗	✗	75.19	74.27
✓	✓	✓	✓	✓	✓	**75.94**	**75.01**

Why Is GDA Robust? From the perspective of Bayesian decision theory[28], the GDA classifier is an approximate implementation of the minimum error rate classifier based on Bayesian theory[29]. When class distributions can be described by Gaussian functions, Bayesian decision theory indicates that the theoretical optimal classification boundary can be achieved using the means and covariances. Noise in labels introduces errors when estimating the class-conditional distributions; however, since GDA employs a global (distribution-level) modeling approach, it approximates the true distribution as a whole. This allows GDA to remain robust and approximate the optimal decision boundary even when some labels are incorrect[29].

5 CONCLUSION

In summary, we propose an innovative test-time adaptation framework for vision-language models by integrating Cache, Residual, and Gaussian. Through residual learning for prototype alignment, Gaussian Discriminant Analysis for category modeling, and a Negative Cache, our method excels under cross-domain and natural distribution shifts, offering a fresh perspective on test-time adaptation.

REFERENCES

[1] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, 2021.

[2] Manli Shu, Weili Nie, De-An Huang, Zhiding Yu, Tom Goldstein, Anima Anandkumar, and Chaowei Xiao. Test-time prompt tuning for zero-shot generalization in vision-language models. *ArXiv*, abs/2209.07511, 2022.

[3] Chun-Mei Feng, Kai Yu, Yong Liu, Salman A. Khan, and Wangmeng Zuo. Diverse data augmentation with diffusions for effective test-time prompt tuning. *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2704–2714, 2023.

[4] Jameel Hassan, Hanan Gani, Noor Hussein, Muhammad Uzair Khattak, Muzammal Naseer, Fahad Shahbaz Khan, and Salman Khan. Align your prompts: Test-time prompting with distribution alignment for zero-shot generalization. *ArXiv*, abs/2311.01459, 2023.

[5] Adilbek Karmanov, Dayan Guan, Shijian Lu, Abdulmotaleb El-Saddik, and Eric P. Xing. Efficient test-time adaptation of vision-language models. *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14162–14171, 2024.

[6] Renrui Zhang, Rongyao Fang, Wei Zhang, Peng Gao, Kunchang Li, Jifeng Dai, Yu Jiao Qiao, and Hongsheng Li. Tip-adapter: Training-free clip-adapter for better vision-language modeling. *ArXiv*, abs/2111.03930, 2021.

[7] Fan Liu, Tianshu Zhang, Wenwen Dai, Wenwen Cai Xiaocong Zhou, and Delong Chen. Few-shot adaptation of multi-modal foundation models: A survey. *ArXiv*, abs/2401.01736, 2024.

[8] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Learning to prompt for vision-language models. *International Journal of Computer Vision*, 130:2337 – 2348, 2021.

[9] Muhammad Uzair Khattak, Hanoona Abdul Rasheed, Muhammad Maaz, Salman H. Khan, and Fahad Shahbaz Khan. Maple: Multi-modal prompt learning. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 19113–19122, 2022.

[10] Peng Gao, Shijie Geng, Renrui Zhang, Teli Ma, Rongyao Fang, Yongfeng Zhang, Hongsheng Li, and Yu Jiao Qiao. Clip-adapter: Better vision-language models with feature adapters. *ArXiv*, abs/2110.04544, 2021.

[11] Christopher Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[12] Ce Zhang, Simon Stepputtis, Katia P. Sycara, and Yaqi Xie. Dual prototype evolving for test-time generalization of vision-language models. *ArXiv*, abs/2410.12790, 2024.

[13] Elaine Sui, Xiaohan Wang, and Serena Yeung-Levy. Just shift it: Test-time prototype shifting for zero-shot generalization with vision-language models. *ArXiv*, abs/2403.12952, 2024.

[14] Prasanta Chandra Mahalanobis. On test and measures of group divergence: theoretical formulae, 1930. 2, 3, 4.

[15] Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*, 2013.

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

[17] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 554–561, 2013.

[18] Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3606–3613, 2013.

[19] Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 12(7):2217–2226, 2019.

[20] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *2008 Sixth Indian conference on computer vision, graphics & image processing*, pages 722–729. IEEE, 2008.

[21] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101–mining discriminative components with random forests. In *Computer vision–ECCV 2014: 13th European conference, zurich, Switzerland, September 6-12, 2014, proceedings, part VI 13*, pages 446–461. Springer, 2014.

[22] Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and CV Jawahar. Cats and dogs. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3498–3505. IEEE, 2012.

[23] Jianxiong Xiao, James Hays, Krista A Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *2010 IEEE computer society conference on computer vision and pattern recognition*, pages 3485–3492. IEEE, 2010.

[24] K Soomro. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.

[25] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[26] Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. Natural adversarial examples. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 15262–15271, 2021.

[27] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do imagenet classifiers generalize to imagenet? In *International conference on machine learning*, pages 5389–5400. PMLR, 2019.

[28] Peter E Hart, David G Stork, Richard O Duda, et al. *Pattern classification*. Wiley Hoboken, 2000.

[29] James O Berger, Elías Moreno, Luis Raul Pericchi, M Jesús Bayarri, José M Bernardo, Juan A Cano, Julián De la Horra, Jacinto Martín, David Ríos-Insúa, Bruno Betrò, et al. An overview of robust bayesian analysis. *Test*, 3(1):5–124, 1994.

[30] Sarah Pratt, Rosanne Liu, and Ali Farhadi. What does a platypus look like? generating customized prompts for zero-shot image classification. *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 15645–15655, 2022.

[31] Hee Suk Yoon, Eunseop Yoon, Joshua Tian Jin Tee, Mark Hasegawa-Johnson, Yingzhen Li, and Chang D Yoo. C-tpt: Calibrated test-time prompt tuning for vision-language models via text feature dispersion. *arXiv preprint arXiv:2403.14119*, 2024.

A APPENDIX

In appendix, we provide additional details and experimental results to enhance understanding and insights into our method. This supplementary document is organized as follows:

- **Detailed Dataset Information:** Comprehensive details about the datasets used in our experiments, including their key characteristics and distributions, are provided.
- **Preliminaries and Methodological Differences:** A discussion of a foundational concept is included, along with an explanation of the key distinctions between our approach and TDA[5].
- **Text Templates for Each Dataset:** The text templates used in our experiments for each dataset are listed for reproducibility.
- **Derivation of Gaussian Discriminant Analysis (GDA):** The mathematical derivation of GDA is detailed.
- **Ablation Study on Hyperparameter $\mathcal{M}$:** An analysis of the hyperparameter M is presented, showing its impact on performance through ablation experiments.
- **Analysis of λ_1 and λ_2:** Insights into the tuning strategies and effects of the hyperparameters λ_1 and λ_2 are discussed.
- **Motivation of $\mathcal{L}_{\text{inter}}^{\text{text}}$ and $\mathcal{L}_{\text{positive}}^{\text{negative}}$:** The motivations behind the losses $\mathcal{L}_{\text{inter}}^{\text{text}}$ and $\mathcal{L}_{\text{positive}}^{\text{negative}}$ are analyzed, demonstrating how they help reduce overconfidence and enhance the model's ability to distinguish positive and negative prototypes.

A.1 DETAILED DATASET INFORMATION

In Table4, we provide comprehensive statistics for each dataset utilized in our experiments, detailing the number of classes, the sizes of the training, validation, and test sets, as well as their associated original tasks. These datasets have emerged as key benchmarks for evaluating the test-time adaptation of vision-language models[2, 13, 5, 3].

A.2 PRELIMINARIES AND METHODOLOGICAL DIFFERENCES

Test-Time Prompt Tuning (TPT). TPT[2] is an augmentation-based adaptation method designed to enhance the generalization of pre-trained models during testing. For each test sample X_{test}, TPT generates n augmented views $\tilde{X}_i = A_i(X_{\text{test}})$ using augmentation functions A, and adapts the model with learned prompts for these views. The objective is to minimize prediction uncertainty by reducing marginal entropy across augmented views while filtering out noisy augmentations through confidence selection. The loss function is defined as:

$$\mathcal{L}_{\text{TPT}} = H\left(\frac{1}{\rho N}\sum_{i=1}^{N}\mathbb{I}\left(H(P(y \mid E(\tilde{X}_i,\theta))) < \tau\right)P(y \mid E(\tilde{X}_i,\theta))\right), \tag{13}$$

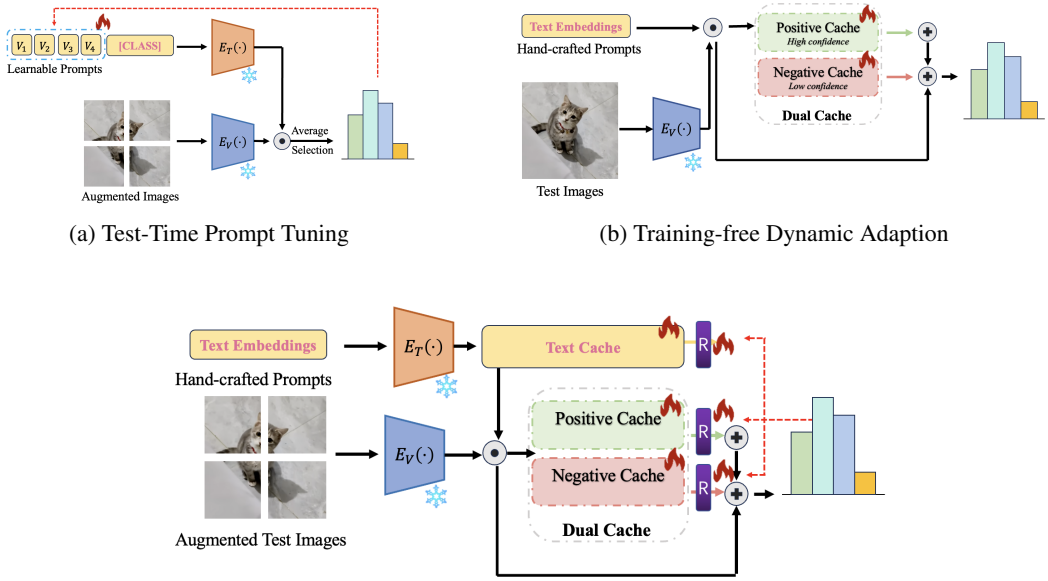
where $P(y \mid E(\tilde{X}_i,\theta))$ is the predicted probability distribution for the augmented view $\tilde{X}_i$, $E(\cdot,\theta)$ is the model with prompt tuning, $H(\cdot)$ is the entropy function, τ is an entropy threshold, and $\mathbb{I}(\cdot)$ is the indicator function ensuring that only augmented views with low entropy are used for training.

It is important to note that TPT provides a new paradigm for optimization, where the specific parameters to be tuned are not fixed but can be adapted flexibly as long as the loss function is followed.

Table 4: Detailed statistics of datasets used in experiments. Note that the last 2 ImageNet variant datasets are designed for evaluation and only contain the test sets. **Datasets marked with an asterisk* indicate that 2,000 samples were randomly selected for testing during the evaluation process.**

Dataset	Classes	Training	Validation	Testing	Task
Caltech101 [16]	100	4,128	1,649	2,465	Object recognition
DTD [18]	47	2,820	1,128	1,692	Texture recognition
EuroSAT [19]	10	13,500	5,400	8,100	Satellite image recognition
FGVCAircraft [15]	100	3,334	3,333	3,333	Fine-grained aircraft recognition
Flowers102 [20]	102	4,093	1,633	2,463	Fine-grained flowers recognition
Food101* [21]	101	50,500	20,200	30,300	Fine-grained food recognition
ImageNet* [25]	1,000	1.28M	-	50,000	Object recognition
OxfordPets [22]	37	2,944	736	3,669	Fine-grained pets recognition
StanfordCars [17]	196	6,509	1,635	8,041	Fine-grained car recognition
SUN397* [23]	397	15,880	3,970	19,850	Scene recognition
UCF101 [24]	101	7,639	1,898	3,783	Action recognition
ImageNet-V2* [27]	1,000	-	-	10,000	Robustness of collocation
ImageNet-A* [26]	200	-	-	7,500	Robustness of adversarial attack

Our method still follows the TPT paradigm. However, unlike the TPT approach, we learn multi-modal residuals, which eliminates the need for gradients to pass through the encoder, significantly improving training speed.



(a) Test-Time Prompt Tuning

(b) Training-free Dynamic Adaption

(c) CRG

Figure 3: Two classic methods of Test-Time Adaptation (top) and our cache-based approach (bottom).

Differences from TDA Although our approach shares many visual similarities with TDA[5] methods, it is fundamentally different in methodology. Our Negative Cache Prototypes are entirely derived from the Positive Cache and are constructed following the principles outlined in the main text. This means we focus exclusively on high-confidence images. Building on this foundation, we further learn a bi-modal residual to better handle each test sample. Additionally, we incorporate Gaussian Discriminant Analysis during inference, which effectively mitigates the impact of noisy labels. The differences between our method and those of TDA and TPT are clearly illustrated in 3

Table 5: Textual prompts used in experiments. In addition to these prompts, we also employ CuPL [30] prompts to further enhance performance.

Dataset	Prompts
ImageNet [25] ImageNet-V2 [27] ImageNet-A [26]	"itap of a {CLASS}." "a bad photo of the {CLASS}." "a origami {CLASS}." "a photo of the large {CLASS}." "a {CLASS} in a video game." "art of the {CLASS}." "a photo of the small {CLASS}."
Caltech101 [16]	"a photo of a {CLASS}."
DTD [18]	"{CLASS} texture."
EuroSAT [19]	"a centered satellite photo of {CLASS}."
FGVCAircraft [15]	"a photo of a {CLASS}, a type of aircraft."
Flowers102 [20]	"a photo of a {CLASS}, a type of flower."
Food101 [21]	"a photo of {CLASS}, a type of food."
OxfordPets [22]	"a photo of a {CLASS}, a type of pet."
StanfordCars [17]	"a photo of a {CLASS}."
SUN397 [23]	"a photo of a {CLASS}."
UCF101 [24]	"a photo of a person doing {CLASS}."

A.3 TEXT TEMPLATES FOR EACH DATASET:

In Table 5, we detail the specific hand-crafted prompts utilized for each dataset.

A.4 DERIVATION OF GAUSSIAN DISCRIMINANT ANALYSIS (GDA)

In the main text, we use Bayes' theorem to compute the posterior probability. By substituting the Gaussian form $p(x \mid y = i) \sim \mathcal{N}(\mu_i, \Sigma)$ into the posterior probability formula, we arrive at the following expression for $p(y = i \mid x)$, which can be represented by a linear classifier. Below, we provide a more detailed illustration of how this posterior probability formula is derived.

Given $p(x \mid y = i) \sim \mathcal{N}(\mu_i, \Sigma)$, the likelihood of class i is

$$p(x \mid y = i) = \frac{1}{(2\pi)^{\frac{d}{2}} |\Sigma|^{\frac{1}{2}}} \exp\Big(-\frac{(x-\mu_i)^T \Sigma^{-1} (x-\mu_i)}{2}\Big), \tag{14}$$

where d is the dimension of the feature vector.

Using Bayes' theorem, the posterior probability $p(y = i \mid x)$ can be written as

$$\begin{aligned}
p(y = i \mid x) &= \frac{p(x \mid y = i)\, p(y = i)}{\sum_{j=1}^{K} p(x \mid y = j)\, p(y = j)} \quad \text{(Bayesian formula)} \\
&= \frac{\frac{1}{(2\pi)^{\frac{d}{2}} |\Sigma|^{\frac{1}{2}}} \exp\Big(-\frac{(x-\mu_i)^T \Sigma^{-1} (x-\mu_i)}{2}\Big) p(y = i)}{\sum_{j=1}^{K} \frac{1}{(2\pi)^{\frac{d}{2}} |\Sigma|^{\frac{1}{2}}} \exp\Big(-\frac{(x-\mu_j)^T \Sigma^{-1} (x-\mu_j)}{2}\Big) p(y = j)} \quad \text{(Using Equation 14)} \\
&= \frac{\exp\Big(\mu_i^T \Sigma^{-1} x - \frac{1}{2} \mu_i^T \Sigma^{-1} \mu_i\Big)\, p(y = i)}{\sum_{j=1}^{K} \exp\Big(\mu_j^T \Sigma^{-1} x - \frac{1}{2} \mu_j^T \Sigma^{-1} \mu_j\Big)\, p(y = j)} \quad \text{(denoted } p(y = i) = p_i\text{)} \\
&= \frac{\exp\big(\mu_i^T \Sigma^{-1} x - \frac{1}{2} \mu_i^T \Sigma^{-1} \mu_i + \log p_i\big)}{\sum_{j=1}^{K} \exp\big(\mu_j^T \Sigma^{-1} x - \frac{1}{2} \mu_j^T \Sigma^{-1} \mu_j + \log p_j\big)},
\end{aligned} \tag{15}$$

where we have omitted constant factors that cancel out in the ratio and combined $\log p(y = i)$ with the exponential term, making the resulting expression directly comparable to a linear classifier.

A.5 ABLATION STUDY ON HYPERPARAMETER $\mathcal{M}$

In our study, the hyperparameter $\mathcal{M}$ plays a crucial role. By default, we set $\mathcal{M}$ to 12, which is significantly larger than the value of 3 commonly used in TDA. This difference arises because Gaussian Discriminant Analysis (GDA) requires calculating the class distribution, particularly the covariance matrix and its inverse. These computations are prone to numerical instability, especially at the beginning of testing, where the classification probabilities calculated by the GDA classifier often result in NaN values.

The robustness of GDA also enables us to use a larger $\mathcal{M}$, as it effectively mitigates the impact of noisy labels. To further investigate, we conducted an analysis on the size of $\mathcal{M}$, and the results are presented below. These experiments were conducted using the RN50 backbone on ImageNet-V2.

Our conclusion is that the size of $\mathcal{M}$ should either be moderately small to maintain the purity of the cache or relatively large to enable GDA to better model the class distributions.

Table 6: Ablation study on the size of $\mathcal{M}$ and its impact on performance on ImageNet-V2.

Size of $\mathcal{M}$	8	10	12	14
Performance (%)	55.6	55.03	**56.07**	55.8

A.6 ANALYSIS OF λ_1 AND λ_2

We perform an ablation study on λ_1 and λ_2. These two parameters are responsible for adjusting the computed positive logits and negative logits, respectively.

As shown in Table 7, on the SUN397 dataset, increasing λ_1 from 5 to 7 slightly improves performance, reaching a peak at 63.11%, while further increasing it to 10 results in a slight performance drop. This indicates that λ_1 requires careful tuning to achieve optimal results. On the other hand, on the Flowers102 dataset, performance increases steadily as λ_2 grows from 0.2 to 0.3, peaking at 71.09%, before slightly dropping at $\lambda_2 = 0.4$. This demonstrates that both λ_1 and λ_2 significantly impact performance. These experiments are conducted using the RN50 backbone.

Table 7: Ablation study on λ_1 and λ_2.

(a) λ_1 Analysis on SUN397

λ_1	5	7	10
Performance (%)	62.96	**63.11**	62.71

(b) λ_2 Analysis on Flowers102

λ_2	0.2	0.3	0.4
Performance (%)	70.84	**71.09**	70.96

A.7 MOTIVATION OF $\mathcal{L}_{\text{INTER}}^{\text{TEXT}}$ AND $\mathcal{L}_{\text{POSITIVE}}^{\text{NEGATIVE}}$

In this section, we discuss our motivation for designing the losses $\mathcal{L}_{\text{inter}}^{\text{text}}$ and $\mathcal{L}_{\text{positive}}^{\text{negative}}$. Specifically, $\mathcal{L}_{\text{inter}}^{\text{text}}$ enforces separations between text prototypes on a hypersphere by minimizing a Gaussian potential kernel G. This approach helps reduce the model's overconfidence during test-time adaptation, thereby lowering the expected calibration error (ECE). Although a prior study [31] suggests that separating text prototypes may degrade performance, however , it brings a slight improvement in performance within our framework. Because our framework differs in two key aspects. First, we anchor the prototypes on a hypersphere by leveraging the Gaussian potential kernel, which provides a smoother separation and better preserves local geometry. Second, our overarching motivation is to control overconfidence stemming from noisy labels, and the corresponding entropy-based Priority Queue cache mechanism effectively filters out high-uncertainty samples. Consequently, once the model's uncertainty is brought under control, the reduction in prediction entropy not only improves calibration but also contributes to slight yet consistent accuracy gains.

Meanwhile, $\mathcal{L}_{\text{positive}}^{\text{negative}}$ explicitly decreases the similarity between positive and negative cached prototypes by accentuating the distinction between "what something is" and "what something is not."

Through this contrastive mechanism, the model gains a clearer understanding of how to discriminate and recognize target concepts, thus further enhancing test-time performance under distribution shifts.