Mitigating Cache Noise in Test-Time Adaptation for Large Vision-Language Models

Anonymous ICME submission

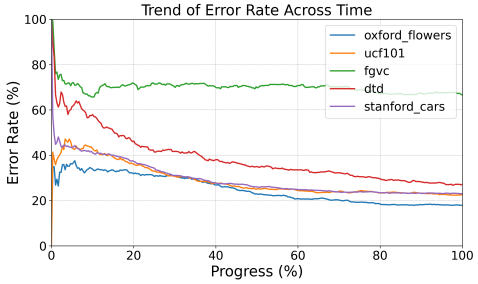
***Abstract*—Test-time adaptation (TTA) of visual language models has recently attracted significant attention as a solution to the performance degradation caused by distribution shifts in downstream tasks. However, existing cache-based TTA methods have certain limitations. They mainly rely on the accuracy 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.**

***Index Terms*—Test-Time Adaptation, Large Vision-Language Model, Feature Alignment**

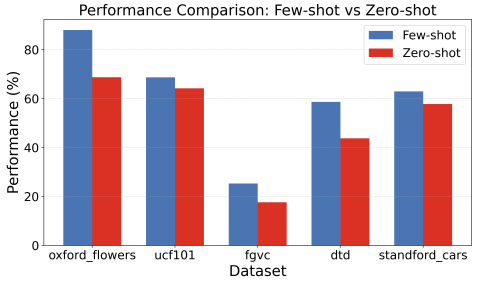
I. Introduction

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]–[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



(a) Error Rate Plot



(b) Few-shot vs Zero-shot

Fig. 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.

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 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 overcome these limitations, we introduce the "Cache, Residual, Gaussian" (CRG) zero-shot test-time adaptation method, which features a comprehensive mechanism to en-

hance cache reliability. This method's overall structure is depicted in Fig. 2. It includes two caches that separately store the positive and negative feature sets of image samples, along with an additional cache dedicated to storing text prototypes. The prototypes for the image features are calculated by averaging the features within the same class. To achieve thorough alignment between text and image prototypes, CRG employs learnable residual vectors, which are optimized by minimizing prediction entropy and maximizing inter-prototype distance.

Despite achieving modality alignment, the cache may still harbor noisy features. To mitigate their effect, we model class distributions using the Gaussian Discriminant Analysis (GDA) framework [7] and utilize Bayes' theorem to calculate posterior probabilities. This decision-making approach, grounded in distribution, effectively minimizes the impact of noisy samples. Additionally, to reduce the overconfidence in predictions induced by noisy features, we construct negative class prototypes that serve as counter-references to further mitigate noise interference. Owing to the synergistic interaction of GDA and negative prototypes, CRG sustains heightened robustness and accuracy even in the presence of noise.

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.

II. Related Works

A. 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 the distribution shift and improve adaptation. Existing CLIP fine-tuning approaches are mainly categorized into two types [8]: prompt-based methods (e.g., CoOp [9] and Maple [10]) and adapter-based methods (e.g., Tip-Adapter [6] and CLIP-Adapter [11]). In this study, we focus on test-time adaptation under zero-shot settings, as acquiring labeled samples in real-world deployments is often impractical.

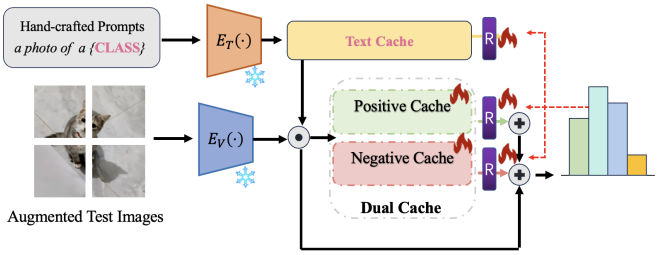


Fig. 2: **Framework Structure of CRG for Zero-Shot Test-time Adaptation** The CRG framework is an extension of CLIP designed for zero-shot online adaptation. It incorporates three cache structures and three learnable residuals, facilitating a comprehensive prediction by integrating features from both CLIP and the cache.

B. 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 the 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.

III. 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.

A. 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.

B. 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.

C. Residual

We observe that residual learning holds significant potential [12] 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.3. 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:

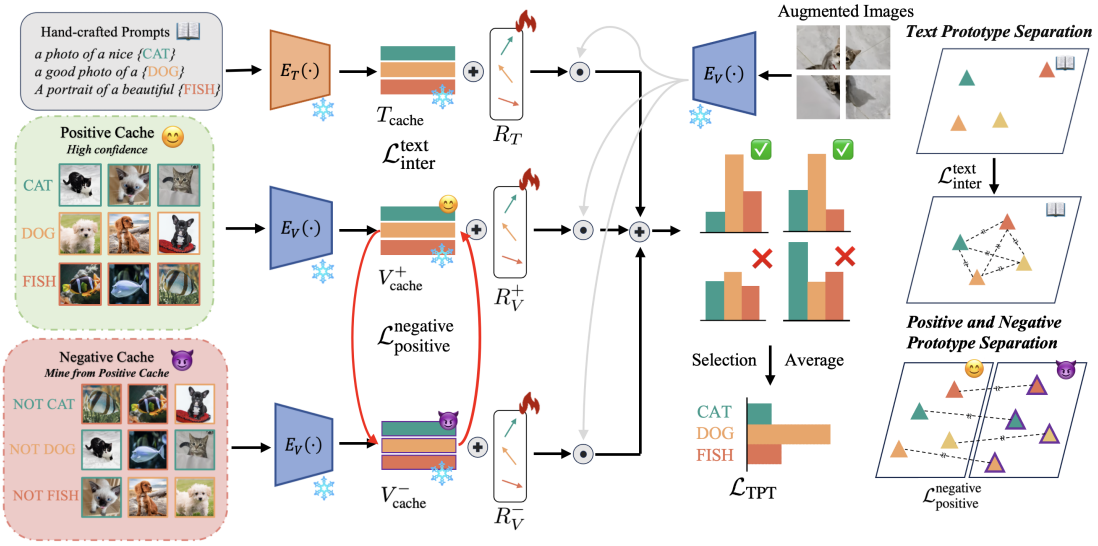


Fig. 3: **The Learning Diagram of CRG**

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

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

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.

D. Gaussian

Theoretical Justification. Recent studies [13] 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 [7]. 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 [7], 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 use the well-aligned positive cache features to compute the mean for each class and the overall covariance, thereby constructing a Gaussian discriminant classifier. 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 the queue of each category.

TABLE I: **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 [9]	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 [9]	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**

Finally, when providing the final inference results, we replace $\mathcal{A}(f_v^\top v_c^+)$ with $\lambda_1 h_c(f_v)$, following the format of Equation 6 and 11, we get Equation 12.

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

IV. Experiment

A. 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, Caltech101, Stanford Cars, DTD, EuroSAT, Flowers102, Food101, Oxford Pets, SUN397, and UCF101. 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 dataset alongside its variant out-of-distribution datasets, including ImageNet-A, ImageNet-V2. 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].

B. 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, 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 II: **Ablation studies for different variants of our method**.

T_{cache}	V_{cache}^+	V_{cache}^-	GDA	$\mathcal{L}_{\text{inter}}^{\text{text}}$	$\mathcal{L}_{\text{positive}}^{\text{negative}}$	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**

C. 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 I, our 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 III, 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.

D. 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 II. Simply adding the

TABLE III: **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 [9]	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 [9]	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**

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.

Why Is GDA Robust? From the perspective of Bayesian decision theory [14], the GDA classifier is an approximate implementation of the minimum error rate classifier based on Bayesian theory [15]. 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 [15].

V. 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*, vol. 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)*, pp. 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*, vol. 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)*, pp. 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*, vol. abs/2111.03930, 2021.

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

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

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

[10] 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)*, pp. 19113–19122, 2022.

[11] 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*, vol. abs/2110.04544, 2021.

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

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

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

[15] 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*, vol. 3, no. 1, pp. 5–124, 1994.