Just Shift It: Test-Time Prototype Shifting for Zero-Shot Generalization with Vision-Language Models

Elaine Sui Xiaohan Wang Serena Yeung-Levy
Stanford University
{esui,xhanwang,syyeung}@stanford.edu

Abstract

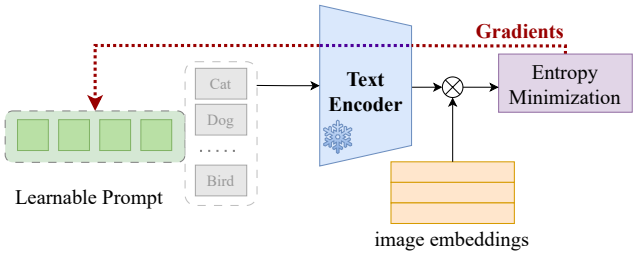
*We introduce the **Test-Time Prototype Shifting (TPS)** framework, a pioneering approach designed to adapt VLMs to test datasets using unlabeled test inputs. Our method involves modulating per-class prototypes generated with a pre-trained text encoder in the shared embedding space by dynamically learning shift vectors for each prototype based solely on the given test sample. This bridges the domain gap and enhances classification accuracy with significantly reduced memory and computational demands compared to state-of-the-art text-prompt tuning methods.*

1. Introduction

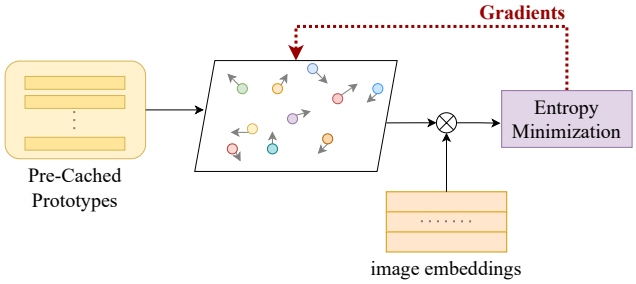
Recently, the field of computer vision has witnessed remarkable progress fueled by the emergence of robust vision-language foundation models [9, 35]. While these models exhibit much better zero-shot generalization compared to ImageNet pre-trained models, they still suffer from performance degradation due to domain shifts at test-time.

In this work, we propose **Test-Time Prototype Shifting** (**TPS**), a simple yet effective framework that specifically adjusts per-class prototypes within the embedding space. Initially, we compute each class prototype using the pre-trained text encoder from a VLM. At test-time, we adapt by learning a shift vector for each prototype on the fly for a single test sample, bridging the domain gap between the prototypes and the target sample. These shift vectors are the only tuneable parameters and are adjusted within the embedding space itself, circumventing the need for backpropagation through the text and visual encoders. Compared to current SoTA method Test-Time Prompt Tuning (TPT) [38], our TPS framework achieves a **10 times** increase in speed while necessitating less than **1/10** of the memory cost.

TPS consistently outperforms CLIP baselines, and surpasses current SoTA by **3.3%** and **1.9%** on the natural distribution shift and cross-dataset generalization benchmarks, respectively. We demonstrate that regardless of the prototype-generation approach, learning a feature-space shift on the prototypes consistently boosts performance over zero-shot CLIP by over **4%** on natural distribution shifts and up to **1%** on cross-dataset generalization benchmarks. Remarkably, our approach not only out-performed TPT in terms of top-1 accuracy but also achieved this with only **1/10** of the memory and time costs.



(a) Test-Time Prompt Tuning



(b) Test-Time Prototype Shifting

Figure 1. Comparison of Test-Time Prompt Tuning (TPT) [38] against our method, Test-Time Prototype Shifting (TPS). TPT requires gradients to backpropagate through the large text encoder to reach the tuneable prompt, incurring high memory and computational costs. In contrast, TPS only backpropagates gradients to the feature space where class prototype shifts are learned, making it much more efficient.

2. Method

2.1. Test-Time Prototype Shifting

Our method comprises three stages: Prototype Generation, Test-Time Shift Tuning, and Test-Time Inference, as depicted in Figure 2. Initially, in the Prototype Generation

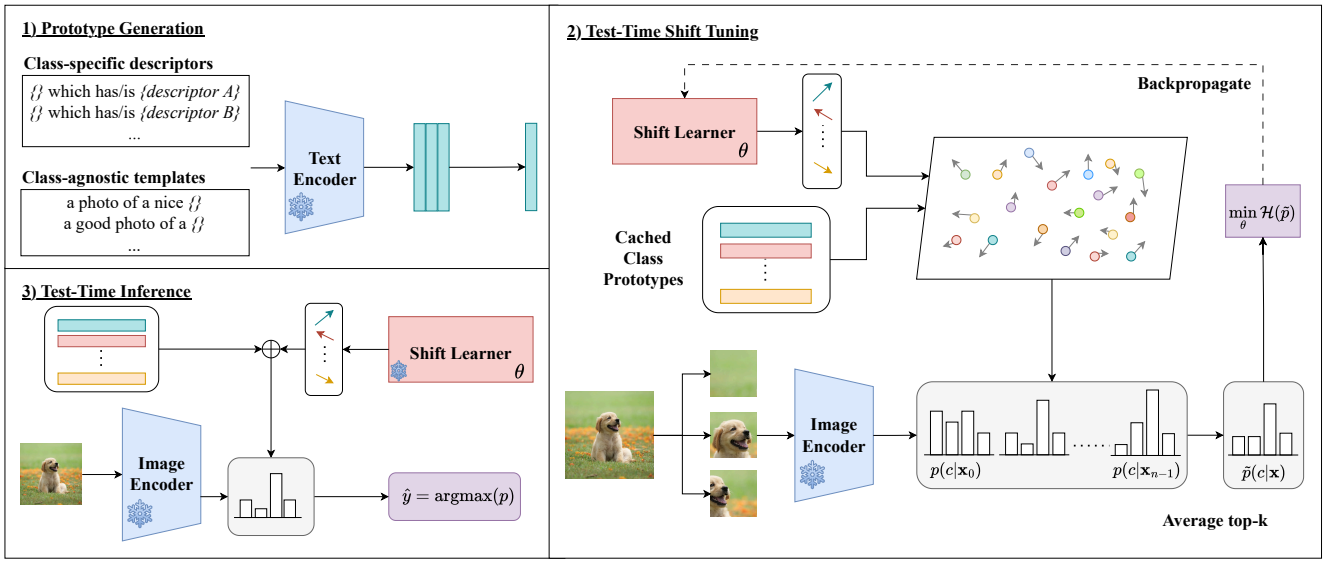


Figure 2. We illustrate the three stages of **Test-Time Prototype Shifting (TPS)**. **1) Prototype Generation:** pre-computation of class prototypes using different prompt-engineering strategies. We show the computation of k class-conditioned descriptors for a single class. Means are computed and cached. **2) Test-Time Shift Tuning:** one iteration of test-time training where we tune the Shift Learner to generate small perturbations to the class prototypes to close the gap between the source and target distributions. Marginal entropy of the CLIP similarities of the shifted prototypes and augmented image embeddings is minimized. **3) Test-Time Inference:** Using the tuned Shift Learner, we compute the final prediction for the shifted class prototypes and the original image embedding with CLIP similarity.

stage, we simply compute the class prototypes by embedding each class template $c \in \mathcal{C}$ of the test dataset. The vanilla prompt is "*a photo of a {class}.*", with more advanced techniques discussed in Section 2.1.2. In the Test-Time Shift Tuning stage, shift vectors are generated through the Shift Learner to modify the prototypes (Section 2.1.1), and cosine similarities between augmented image embeddings and shifted class prototypes are used to produce n probability distributions. We optimize the Shift Learner to minimize the entropy of the aggregated marginal distribution. Finally, in the Test-Time Inference stage (Section 2.1.3), we predict the class by comparing the cosine similarities between the learned shifted text embeddings and the original image embedding, choosing the class with the highest probability.

2.1.1 Feature-Space Shift

We observe that prompt tuning, when used for test-time adaptation, acts as an indirect technique for adjusting text embeddings to bridge the domain gap, simultaneously leveraging the multi-modal embedded knowledge of CLIP. So, why not directly modulate class prototypes within the embedding space? This inspired the development of our Shift Learner module, designed specifically for learning small perturbations to the text embeddings. This approach not only capitalizes on the exceptional quality of the CLIP representation space but also adeptly avoids the requirement for the computationally demanding gradient backpropagation through the text encoder. We propose to learn shifts for specific class prototypes as domain gaps often consist of both dataset-level shifts (e.g. transitioning from natural images to sketches) and class-level distribution shifts (e.g. differences between indoor and outdoor settings for dogs).

We formally elaborate on the operation of our per-class shift as follows, given class $c \in \mathcal{C}$ and corresponding class prototype $\mathbf{p}_c \in \mathbb{R}^d$, we learn a shift vector $\mathbf{s}_c \in \mathbb{R}^d$. The shift operation is performed by channel-wise addition between the class prototype $\mathbf{p}_c$ and the learned shift vector $\mathbf{s}_c$. The normalized shifted prototype $\mathbf{p'}_c \in \mathbb{R}^d$ is generated as

$$\mathbf{p'}_c = \frac{\mathbf{p}_c + \mathbf{s}_c}{||\mathbf{p}_c + \mathbf{s}_c||_2} \tag{1}$$

2.1.2 Advanced Prototype Generation

By learning shifts on pre-computed and cached class prototypes, TPS is designed to be seamlessly compatible with any existing prompt-engineering methods as a plug-and-play module of our framework. This integration empowers the generation of more robust and effective prototypes, leveraging advanced prompting strategies and offline prototype adjustment. In this work, we employ three distinct prompt techniques: class-agnostic prompts using the set of 80 hand-crafted templates provided by CLIP [35] for ImageNet, class-specific prompts derived from intricate class descriptions generated by GPT-4 [31], and learnable prompts such as CoOp [49] and CoCoOp [50].

Following [27, 35], we can easily improve the robustness of class prototypes by taking the mean of the class-specific embeddings. This allows us to leverage multiple prompting and prompt-learning techniques and retain the knowledge from these various representations while maintaining the computational and memory efficiency of our method. We combine the class-agnostic prompts and per-class descriptors to generate the final prototype set $\{\mathbf{p}_c\}_{c\in\mathcal{C}}$. Several types of combinations are discussed in the Appendix.

2.1.3 Test-Time Training and Inference

At test time, given a single test image v_0, we follow [38] to augment it $(n-1)$ times and compute the features of the original and augmented images with the CLIP image encoder to obtain embeddings $\{\mathbf{x}_i\}_{i=0}^{n-1}$. As introduced in Section 2.1.1, we shift the pre-cached prototypes $\{\mathbf{p}_c\}_{c\in\mathcal{C}}$ to obtain $\{\mathbf{p}'_c\}_{c\in\mathcal{C}}$. For each image feature $\mathbf{x}_i$, the predicted probabilities are calculated as

$$p(c|\mathbf{x}_i, \mathbf{p}'_c) = \frac{\exp\left(\mathbf{p}'^{\top}_c \mathbf{x}_i/\tau\right)}{\sum_{c\in C}\exp\left(\mathbf{p}'^{\top}_c \mathbf{x}_i/\tau\right)} \tag{2}$$

where τ is the temperature scalar. Similar to TPT [38], we select the k distributions with highest confidence (**i.e**. lowest entropy) of the batch and take the average. Denoting the image embeddings corresponding to the selected k distributions as $\{\mathbf{x}'_i\}_{i=1}^{k}$, we train our model to minimize the following entropy of this marginal distribution,

$$\mathcal{L} = -\sum_{c\in\mathcal{C}} \tilde{p}(c|\mathbf{x}_0, \mathbf{p}'_c)\log\tilde{p}(c|\mathbf{x}_0, \mathbf{p}'_c) \tag{3}$$

$$\text{where } \tilde{p}(c|\mathbf{x}_0) = \frac{1}{k}\sum_{i=1}^{k} p(c|\mathbf{x}'_i, \mathbf{p}'_c) \tag{4}$$

In our model, the only parameters that are optimized are the shift vectors $\{\mathbf{s}_c\}_{c\in\mathcal{C}}$. We update the shift vectors for a single step of gradient descent.

After test-time training, we encode the original image and compute its cosine similarity with the shifted class prototypes, resulting in a final prediction that is the `argmax` of the prediction logits. Algorithm 1 in the Appendix summarizes the entire procedure of our proposed method, TPS, that enables efficient test-time adaptation using VLMs.

3. Experimental Results

3.1. Datasets

We evaluate our method TPS on natural distributions shifts and cross-dataset generalization. For natural distribution shifts, we evaluate ImageNet [6] along with its four variants: ImageNet-V2 [36], ImageNet-A [15], ImageNet-R [14] and ImageNet-Sketch [44]. For cross-dataset generalization, we evaluate on: Flowers102 [28], DTD[4], OxfordPets [32], StanfordCars [19], UCF101 [40], Caltech101 [7], Food101 [1], SUN397 [45], FGVC-Aircraft [26], and EuroSAT [13]. We report Top-1 accuracy for image classification on all datasets.

Method	ImageNet	ImageNet OOD Avg	Cross-Dataset Avg
	Zero-Shot Baseline		
CLIP-ViT-B/16	66.74	57.20	63.45
	Test-Time Adaption Baselines		
TPT [38]	68.98	60.81	65.10
TPT [38] + descriptors*	67.71	58.75	64.77
TPT + (templates + descriptors)*	69.54	61.22	65.16
DiffTPT [8]	70.30	60.52	65.47
Ours (Shift + templates + descriptors)	**71.45**$_{(\uparrow 4.71)}$	**64.15**$_{(\uparrow 6.95)}$	**66.96**$_{(\uparrow 3.51)}$

Table 1. Acc@1 of zero-shot image classification with CLIP-ViT-B/16 backbone. Performance improvements over zero-shot CLIP are denoted in (↑blue). Best performances are in **bold**.

Method	ImageNet	Average	OOD Average
	Zero-Shot Baseline		
CLIP-ViT-B/16 + CoOp [49]	71.51	61.72	59.28
	Test-Time Adaption Baselines		
TPT + CoOp ([38, 49])	73.61	64.99	62.83
DiffTPT + CoOp ([8, 49])	**75.00**	64.12	61.97
Ours (Shift + CoOp[49])	73.73$_{(\uparrow 2.22)}$	**65.52**$_{(\uparrow 3.80)}$	**63.46**$_{(\uparrow 4.18)}$

Table 2. Acc@1 of zero-shot image classification with CLIP-ViT-B/16 backbone on ImageNet and its OOD variants using CoOp-learned prompts. Performance improvements over zero-shot CLIP are denoted in (↑blue). Best performances are in **bold**.

3.2. Comparison to State-of-the-Art

3.2.1 Baselines

We compare our method with zero-shot and TTA baselines that leverage CLIP ViT-B/16. To make our method more comparable to the simplest baselines, we also augment them by includign of class-agnostic CLIP templates [35] (**+ templates**), class-specific LLM-generated descriptors (**+ descriptors**), and learned CoOp [49] prompts (**+ CoOp**).

As Test-Time Prompt Tuning (TPT) does not involve tuning in the feature space, we append descriptors to the prompt suffixes, denoted as **+ descriptors***. To further augment the TPT-tuned class prototypes, we take its mean with the same advanced prototypes used at initialization in our method, denoted as **+(templates + descriptors)***.

3.2.2 Results

Table 1 presents the top-1 accuracy of TPS, benchmarked against zero-shot and test-time adaptation (TTA) baselines using CLIP. Our results demonstrate that shifting class prototypes significantly enhances performance. Compared to the baseline zero-shot CLIP, we observe an improvement of 7%, and a 3.3% increase over the vanilla TPT on average for ImageNet out-of-distribution datasets. In addition, we observe an average improvement of 3.5% over the zero-

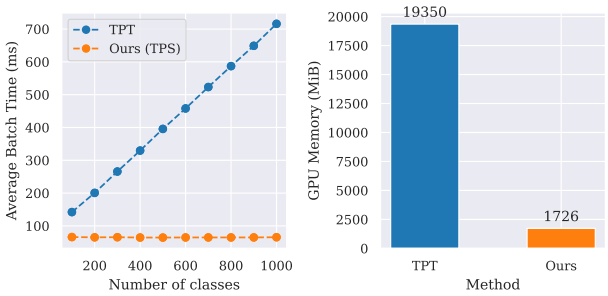


Figure 3. Comparison of computational and memory costs on an A6000 GPU on ImageNet. **Left:** Average runtimes of TPT and TPS across different sized subsets of ImageNet [6] over 3 runs. Note: error bars are depicted but not visible as they have extremely small standard deviations. **Right:** Memory consumption of TPT and TPS on ImageNet.

shot CLIP, and a 1.9% increase compared to TPT for the cross-dataset generalization benchmarks.

Table 1 also shows that directly appending descriptors to the TPT prompt suffixes results in a performance decrease of 2%, emphasizing the limitations of TPT in seamlessly incorporating prompt-engineering techniques. Notably, Table 2 demonstrates that our approach of learning a feature-based shift outperforms TPT and DiffTPT by 0.6% and 1.5%, respectively, on average even when using advanced prototypes derived from learned CoOp [49] prompts without backpropagating through the text encoder or prompting a diffusion model. This finding underscores that feature space modulation can effectively replicate the impact of test-time prompt tuning in scenarios involving natural distribution shifts. Full results are shown in Appendix § C.

3.3. Efficiency Analysis

Test-time adaptation involves tuning a model on an out-of-distribution dataset at test-time given a single input or batch at a time. As the input is streaming, each adaptation requires low memory and computational cost to be used in practice.

Figure 3 shows the average batch runtime and GPU memory consumption on ImageNet for TPS and the simplest TTA baseline TPT on a single A6000 GPU. We show the runtime of TPT and TPS on subsets of ImageNet comprised of different sized label sets. We observe that the average runtime per test input for TPT scales linearly with the size of the label set while our method, TPS, remains constant at approximately 65 ms per batch on average. On the full ImageNet test set, we further see that TPS runs **more than 10x as fast** and uses **less than 10x the memory** as TPT, and yet is able to achieve performance gains over the baseline. These significant speed-ups and low memory constraints highlight the cost of backpropagating through the text encoder and illustrate the practicality of TPS.

Prompt Type	Setting	ImageNet (IN)	IN-OOD Avg	Cross-Dataset Avg
Vanilla	Zero-Shot	66.74	57.20	63.45
	+ shift	68.77	61.59	64.41
	Δ	**+ 2.03**	**+ 4.39**	**+ 0.96**
CoOp [49]	Zero-Shot	71.51	59.28	N/A
	+ shift	73.73	63.46	N/A
	Δ	**+ 2.22**	**+ 4.18**	N/A
CLIP templates	Zero-Shot	68.35	59.43	64.69
	+ shift	70.38	64.04	65.57
	Δ	**+ 2.03**	**+ 4.61**	**+ 0.88**
Descriptors	Zero-Shot	68.52	58.29	66.02
	+ shift	70.40	62.48	66.80
	Δ	**+ 1.88**	**+ 4.19**	**+ 0.78**
CLIP templates + Descriptors	Zero-Shot	69.54	59.88	65.94
	+ shift	71.45	64.15	66.96
	Δ	**+ 1.91**	**+ 4.27**	**+ 1.02**

Table 3. Acc@1 for zero-shot image classification comparing pure zero-shot baselines vs. with learned feature-space shift from prototypes derived from various prompts using CLIP-ViT/B-16.

Method	ImageNet (IN)	IN-OOD Avg	Cross-Dataset Avg
Zero-shot	76.28	72.20	73.72
Ours (TPS)	**77.66**	**75.75**	**74.16**

Table 4. Acc@1 for zero-shot classification with CLIP-ViT-L/14.

3.4. Effect of Shift on Different Prototypes

We explore the effect of feature-space shift on a variety of prototypes. Specifically, we compare our method TPS against zero-shot performance given the same prototypes constructed from the vanilla *"a photo of a {class}"* prompt, CoOp [49]-learned prompt, the 80 ImageNet context prompts from CLIP [35] and our LLM-generated descriptors. As observed in Table 3, regardless of the prototype generation technique used, introducing minor perturbations to class prototypes consistently yields an average gain of $> 4\%$ in top-1 accuracy on ImageNet OOD datasets and up to 1% on cross-domain datasets over zero-shot CLIP with the same prototypes. This illustrates how the embedding space structured is maintained with a learnable shift.

3.5. Larger Backbone

Table 4 illustrates our results using CLIP-ViT-L/14. We demonstrate that TPS outperforms zero-shot CLIP for both natural distribution shifts and cross-dataset generalization.

4. Conclusion

In this work, we present the Test-time Prototype Shifting (TPS) framework, a novel approach to enhance the zero-shot generalization abilities of VLMs. TPS addresses the limitations of existing test-time training methods by directly modulating class prototypes in the embedding space. This strategy not only reduces the computational and memory demands significantly but also allows for greater flexibility and precision in adapting to diverse domain shifts.

References

[1] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101 – mining discriminative components with random forests. In **European Conference on Computer Vision**, 2014. 3

[2] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In **NeurIPS**, 2020. 1

[3] Ting Chen, Mario Lucic, Neil Houlsby, and Sylvain Gelly. On self modulation for generative adversarial networks. In **International Conference on Learning Representations**, 2019. 1

[4] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, , and A. Vedaldi. Describing textures in the wild. In **Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)**, 2014. 3

[5] Harm de Vries, Florian Strub, Jeremie Mary, Hugo Larochelle, Olivier Pietquin, and Aaron C Courville. Modulating early visual processing by language. In **Advances in Neural Information Processing Systems**, 2017. 1

[6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In **CVPR**, 2009. 3, 4

[7] 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 **CVPR Workshops**, 2004. 3

[8] Chun-Mei Feng, Kai Yu, Yong Liu, Salman Khan, and Wangmeng Zuo. Diverse data augmentation with diffusions for effective test-time prompt tuning. In **Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)**, pages 2704–2714, 2023. 3, 1

[9] Rohit Girdhar, Alaaeldin El-Nouby, Zhuang Liu, Mannat Singh, Kalyan Vasudev Alwala, Armand Joulin, and Ishan Misra. Imagebind: One embedding space to bind them all. In **Conference on Computer Vision and Pattern Recognition (CVPR)**, 2023. 1

[10] Sachin Goyal, Mingjie Sun, Aditi Raghunathan, and J. Zico Kolter. Test time adaptation via conjugate pseudo-labels. In **Advances in Neural Information Processing Systems**, pages 6204–6218, 2022. 1

[11] Ziyu Guo, Renrui Zhang, Longtian Qiu, Xianzheng Ma, Xupeng Miao, Xuming He, and Bin Cui. Calip: Zero-shot enhancement of clip with parameter-free attention. In **AAAI**, 2023. 1

[12] 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. In **Thirty-seventh Conference on Neural Information Processing Systems**, 2023. 1

[13] 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 J. Sel. Top. Appl. Earth Obs. Remote. Sens.**, 2019. 3

[14] Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, Dawn Song, Jacob Steinhardt, and Justin Gilmer. The many faces of robustness: A critical analysis of out-of-distribution generalization. **ICCV**, 2021. 3

[15] 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 (CVPR)**, pages 15262–15271, 2021. 3

[16] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In **ICCV**, 2017. 1

[17] Yusuke Iwasawa and Yutaka Matsuo. Test-time classifier adjustment module for model-agnostic domain generalization. In **Advances in Neural Information Processing Systems**, 2021. 1

[18] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and koray kavukcuoglu. Spatial transformer networks. In **Advances in Neural Information Processing Systems**, 2015. 1

[19] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In **ICCV Workshops**, 2013. 3

[20] Dongze Lian, Daquan Zhou, Jiashi Feng, and Xinchao Wang. Scaling & shifting your features: A new baseline for efficient model tuning. In **Advances in Neural Information Processing Systems (NeurIPS)**, 2022. 1

[21] Wei Lin, Muhammad Jehanzeb Mirza, Mateusz Kozinski, Horst Possegger, Hilde Kuehne, and Horst Bischof. Video test-time adaptation for action recognition. In **CVPR**, 2023. 1

[22] Shihong Liu, Samuel Yu, Zhiqiu Lin, Deepak Pathak, and Deva Ramanan. Language models as black-box optimizers for vision-language models, 2023. 1

[23] Yuejiang Liu, Parth Kothari, Bastien van Delft, Baptiste Bellot-Gurlet, Taylor Mordan, and Alexandre Alahi. Ttt++: When does self-supervised test-time training fail or thrive? In **Advances in Neural Information Processing Systems**, pages 21808–21820. Curran Associates, Inc., 2021. 1

[24] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In **International Conference on Learning Representations**, 2019. 1

[25] Xiaosong Ma, Jie ZHANG, Song Guo, and Wenchao Xu. Swapprompt: Test-time prompt adaptation for vision-language models. In **Thirty-seventh Conference on Neural Information Processing Systems**, 2023. 1

[26] S. Maji, J. Kannala, E. Rahtu, M. Blaschko, and A. Vedaldi. Fine-grained visual classification of aircraft. Technical report, 2013. 3

[27] Sachit Menon and Carl Vondrick. Visual classification via description from large language models. **ICLR**, 2023. 3, 1

[28] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In **Indian Conference on Computer Vision, Graphics and Image Processing**, 2008. 3

[29] Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Yaofo Chen, Shijian Zheng, Peilin Zhao, and Mingkui Tan. Efficient test-time model adaptation without forgetting. In **ICML**, 2022. 1

[30] Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Zhiquan Wen, Yaofo Chen, Peilin Zhao, and Mingkui Tan. Towards stable test-time adaptation in dynamic wild world. **ICLR**, 2023. 1

[31] OpenAI. Gpt-4 technical report, 2023. 2, 4

[32] Omkar M. Parkhi, Andrea Vedaldi, Andrew Zisserman, and C. V. Jawahar. Cats and dogs. In **CVPR**, 2012. 3

[33] Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron C. Courville. Film: Visual reasoning with a general conditioning layer. In **AAAI**, 2018. 1, 2

[34] Mihir Prabhudesai, Tsung-Wei Ke, Alexander C. Li, Deepak Pathak, and Katerina Fragkiadaki. Test-time adaptation of discriminative models via diffusion generative feedback. In **Conference on Neural Information Processing Systems**, 2023. 1

[35] 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 **ICML**, 2021. 1, 2, 3, 4

[36] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do ImageNet classifiers generalize to ImageNet? In **Proceedings of the 36th International Conference on Machine Learning**, pages 5389–5400. PMLR, 2019. 3

[37] Steffen Schneider, Evgenia Rusak, Luisa Eck, Oliver Bringmann, Wieland Brendel, and Matthias Bethge. Improving robustness against common corruptions by covariate shift adaptation. In **NeurIPS**, 2020. 1

[38] 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. In **Advances in Neural Information Processing Systems**, pages 14274–14289. Curran Associates, Inc., 2022. 1, 3, 2, 4

[39] Junha Song, Jungsoo Lee, In So Kweon, and Sungha Choi. Ecotta: Memory-efficient continual test-time adaptation via self-distilled regularization. In **CVPR**, 2023. 1

[40] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. **CoRR**, abs/1212.0402, 2012. 3

[41] Yu Sun, Xiaolong Wang, Liu Zhuang, John Miller, Moritz Hardt, and Alexei A. Efros. Test-time training with self-supervision for generalization under distribution shifts. In **ICML**, 2020. 1

[42] Vishaal Udandarao, Ankush Gupta, and Samuel Albanie. Sus-x: Training-free name-only transfer of vision-language models. In **ICCV**, 2023. 1

[43] Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno A. Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. In **ICLR**, 2021. 1

[44] Haohan Wang, Songwei Ge, Zachary Lipton, and Eric P Xing. Learning robust global representations by penalizing local predictive power. In **Advances in Neural Information Processing Systems**, pages 10506–10518, 2019. 3

[45] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In **2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition**, 2010. 3

[46] Yue Yang, Artemis Panagopoulou, Shenghao Zhou, Daniel Jin, Chris Callison-Burch, and Mark Yatskar. Language in a bottle: Language model guided concept bottlenecks for interpretable image classification. In **CVPR**, 2023. 1

[47] Marvin Zhang, Sergey Levine, and Chelsea Finn. MEMO: test time robustness via adaptation and augmentation. **CoRR**, abs/2110.09506, 2021. 1

[48] Shuai Zhao, Xiaohan Wang, Linchao Zhu, and Yi Yang. Test-time adaptation with clip reward for zero-shot generalization in vision-language models. In **ICLR**, 2024. 1

[49] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Learning to prompt for vision-language models. **CoRR**, abs/2109.01134, 2021. 2, 3, 4, 1, 5

[50] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Conditional prompt learning for vision-language models. In **CVPR**, 2022. 2, 1

[51] Yifei Zhou, Juntao Ren, Fengyu Li, Ramin Zabih, and Ser-Nam Lim. Test-time distribution normalization for contrastively learned visual-language models. In **Thirty-seventh Conference on Neural Information Processing Systems**, 2023. 1

This document provides more details of our approach and additional experimental results, organized as follows:

- § A Related Works
- § B Implementation Details
- § C Additional Quantitative Results with Different Random Seeds
- § D Additional Ablation Studies
- § E Research Impact and Limitations

A. Related Works

A.1. Test-Time Adaptation

Test-time adaptation (TTA) is the task of adapting a model's weights on an unlabeled out-of-distribution test set in order to achieve higher test performance. In the context of vision tasks, traditional methods leverage ImageNet-pretrained image classifiers and use techniques such as computing pseudo-prototype class representations to update the linear classifier [17], learning better feature representations through self-supervised auxiliary tasks [21, 23, 41], adapting the normalization layers to learn the statistics of the target distribution [29, 30, 37, 39, 43], as well as minimizing prediction entropy to increase the confidence of predictions [10, 29, 30, 43, 47]. With the development of CLIP [35], recent work in TTA has been predominantly based on prompt tuning. This involves learning a tuneable text and/or image prompt to encode the visual distribution shift while maintaining the strong performances of these foundation models by keeping the pre-trained parameters fixed [8, 12, 25, 38, 48]. Despite only tuning a relatively small number of prompt parameters, tuning the input requires backpropagating through their respective encoders, which is especially memory intensive with large input sizes, making it infeasible in practice. Recently, an image diffusion-based method DiffusionTTA [34] has been proposed for TTA, involving updating classifier weights by training for the auxiliary task of image reconstruction using a conditional diffusion model. Nevertheless, this also necessitates backpropagation through the classifier and diffusion model. In contrast, this work proposes to avoid backpropagation through the encoders and maintain the richness of the CLIP embedding space by directly modulating the features within it.

Current TTA methods also comprise those that are considered training-free [11, 42]. Approaches include adding a parameter-free attention module to modulate multi-modal features [11] and computing the similarity between the target image and those from a constructed support set [42]. Although our work does not directly fit this setting, we follow the same spirit of minimally adjusting intermediate representations to close the domain gap.

A.2. Feature Modulation

Feature modulation is a parameter-efficient tuning paradigm where features are perturbed to better conform to a target task. This learned perturbation is typically in the form of feature normalization, achieved by modulating the encoder's normalization layers to align source and target tasks [3, 5, 16, 18, 33]. However, modulation can also be applied more directly to the features themselves [20, 51]. For example, SSF [20] proposes to learn scale and shift parameters for each layer's activations. DN [51] proposes to subtract the means of the text and image embeddings from the respective inputs before computing CLIP similarity to align the CLIP training and inference procedures. We propose a more simplistic feature modulation procedure where we only learn shift vectors to pre-computed class prototypes in test-time training to better align them with the out-of-distribution image embeddings of the target dataset.

A.3. Prompting for Vision-Language Models

Vision-language models enable zero-shot generalization to downstream datasets via prompting. As predictions are computed by cosine similarity of the text and image embeddings, the quality of the text embeddings or class prototypes can cause a drastic difference in performance. In the case of image classification, this entails the careful design of natural language text descriptions for each of the class names, focusing on the visual aspects apparent from the image itself [35]. CoOp [49] removes the need for hand-crafting prompts by prompt-tuning in the few-shot setting and CoCoOp [50] extends CoOp by learning instance-conditioned prompts, improving generalization ability. Another paradigm of prompt-engineering includes prompting large language models (LLMs) for better prompt templates [22] and/or content [27, 46]. Specifically, Menon and Vondrick [27] and Yang **et al**. [46] both prompt GPT-3 [2] to generate concepts or descriptors of class names to increase zero-shot and linear-probe performance on image classification while providing model interpretability. Our work leverages these developments in prompt-engineering in our prototype generation phase, using these techniques to generate more knowledge-rich prototypes that can be swapped into the framework in a plug-and-play manner.

B. Implementation Details

Similar to TPT [38], we augment a test image 63 times with random resized crops to obtain a batch of 64 images that also includes the original image. We select 10% of samples in the batch with lowest entropy and compute the marginal entropy of the selected predicted probability distributions. We initialize the learnable shift to all zeros and similarly to TPT [38], optimize it for 1 step using the AdamW [24] optimizer and learning rate of 5e-3 for ImageNet variants

and 1e-3 for cross-dataset generalization, which we found by hyperparameter tuning on the validation set. For our method, we initialize each class prototype by taking the micro average of the mean of the class-agnostic CLIP template prompts and the mean of the class-specific GPT-4 generated descriptive prompts.

B.1. Detailed Pseudocode

Algorithm 1 shows more detailed pseudocode in PyTorch-like style for Test-Time Prototype Shifting over an entire dataset. We will release the models and source code to ensure reproducibility.

C. Main Results With More Random Seeds

In Sec C.1 and C.2, we run Test-Time Prototype Shifting (TPS) over 3 random seeds on both the natural distribution shifts and cross-dataset generalization (Table 1), respectively. The randomness comes from the image augmentation in creating a diverse minibatch for the entropy minimization objective.

C.1. Natural Distribution Shifts

From Table 7, we observe that our conclusion from Sec 3.2.2 still holds. That is, our method outperforms SoTA TPT [38] by $> 3.4\%$ on average. We also observe that augmenting the TPT-tuned class prototypes with more advanced off-the-shelf prototypes only boosts performance by a mere 0.5% on average over vanilla TPT, demonstrating TPT's limitation in maximally leveraging these advanced prototypes.

C.2. Cross-Dataset Generalization

From Table 8, we see that our conclusion from Sec 3.2.2 remains valid. Specifically, TPS outperforms TPT [38] by $> 2\%$ on average. Similarly to Sec C.1, we observe that taking the mean of the TPT-tuned and advanced off-the-shelf prototypes increases performance by only 0.5% on average over TPT, demonstrating TPT's inflexibility in utilizing these more robust class representations.

D. Full Ablations

In Sec D.1, we compare different types of feature-space transformations, motivating our choice of shifting features. In Sec D.4, we report full ablations on TPS on the effectiveness of feature-space shift on various prototypes. These results are comparable to those reported in Sec 3.4. In Sec D.5, we include additional ablations to observe the effect of learning a class-specific shift over a universal shift for all classes. In Sec D.6, we explore variants on prototype generation using the class-agnostic CLIP ImageNet context prompt templates [35] and the class-specific descriptors generated using GPT-4 [31].

D.1. Feature-Space Transformation Variants

We compare different variants for feature-space transformations in Table 9. Specifically, we compare against element-wise *scale*, element-wise *scale&shift*, as well as *FiLM* [33] where affine vectors are computed via a linear layer on the prototypes. We observe that *scale* performs worse than *shift*, and *scale&shift* performs equally to *shift*. On the other hand, *FiLM* suffers from model collapse due to much more learnable weights in TTA. Considering both efficiency and efficacy, *shift* is the best choice in our framework.

D.2. Analysis of Number of Gradient Steps

We show the results of adapting for more than one gradient step in Table 10. We observe that a single TTA step is sufficient to achieve the performance gains and that performance plateaus as the number of steps increases.

D.3. Larger Backbone

We include the full results of using a CLIP-ViT-L/14 backbone in Tables 11 and 12. We demonstrate that our conclusion from Sec 3.5 still holds, that learning small feature-space shifts improves performance regardless of model scale.

D.4. Effect of Shift on Different Prototypes

Full comparisons between zero-shot and feature-shifted performance on all natural distribution shift and cross-domain generalization benchmark datasets over 3 random seeds are in Tables 13 and 14, respectively. We demonstrate that our conclusion from Sec 3.4 stills holds – that learning a small perturbation in the feature space results in performance gains of $> 4\%$ and up to 1% on average across natural distribution shift and cross-domain generalization tasks regardless of what prototypes are used.

D.5. Effect of Per-Class vs. Shared Shift

Test-time prompt tuning methods involve tuning a prompt that is shared across all classes in a dataset. Given that the tuneable prompt tokens form a portion of the text encoder input, these full prompts are then mapped to the embedding space with the encoder's learned complex feature-space mapping. This results in non-linear perturbations from the original class prototypes. However, for our method, tuning shift parameters that are shared for all class prototypes in the feature-space means that the relative distance between class prototypes will remain constant before and after test-time shift tuning, limiting the expressive capability of the learned shift. Rather, we believe that each class prototype should be modulated by slightly different magnitudes and/or directions to provide more degrees of freedom in capturing the class-level distribution shifts in addition to the dataset-level shifts present in a domain gap.

Algorithm 1 Test-Time Prototype Shifting Pseudocode in PyTorch-like style

```
# Define frozen parameters
image_encoder = CLIPImageEncoder()
prototypes = load_class_prototypes()

predictions = []
for img, label in data_loader:
    # Test-Time Shifting
    shift_params = nn.Parameter(torch.zeros(num_classes, embed_dim), requires_grad=True)
    aug_imgs = [aug(img) for i in range(batch_size - 1)]
    imgs = torch.stack([img] + aug_imgs, dim=0)
    image_features = image_encoder(imgs)

    text_features = prototypes + shift_params
    text_features = F.normalize(text_features, dim=-1)

    logits = (logit_scale * text_features @ image_features.T)

    # Confidence selection
    entropies = compute_batch_entropies(logits)
    top_k_idx = torch.argsort(batch_entropy, descending=False)[:k]

    loss = compute_average_entropy(logits[top_k_idx])
    optimizer.zero_grad()
    loss.backward()
    optimizer.step()

    # Test-Time Inference
    new_prototypes = prototypes + shift_params
    new_prototypes = F.normalize(new_prototypes, dim=-1)

    logits = (logit_scale * new_prototypes @ image_features[0].unsqueeze(0).T)
    pred = torch.argmax(logits)

    predictions.append(pred)

return predictions
```

Method	ImageNet	ImageNet-A	ImageNet-V2	ImageNet-R	ImageNet-Sketch	Average	OOD Average
	Test-Time Adaptation Baselines						
TPT [38]	68.96 (±.03)	54.47 (±.26)	63.46 (±.07)	77.10 (±.04)	47.93 (±.03)	62.38 (±.05)	60.74 (±.06)
TPT + (templates + descriptors)*	69.51 (±.05)	54.94 (±.17)	63.86 (±.11)	77.57 (±.11)	48.38 (±.04)	62.85 (±.03)	61.19 (±.04)
Ours	**71.43** (±.06)	**60.78** (±.21)	**65.00** (±.09)	**80.06** (±.13)	**50.97** (±.09)	**65.65** (±.06)	**64.20** (±.08)

Table 7. Acc@1 of zero-shot image classification with CLIP-ViT-B/16 backbone on ImageNet and its OOD variants over 3 random seeds. Best performances are in **bold**.

Table 15 shows that, on average, learning a per-class shift increases performance by $> 1.2\%$ regardless of which prototypes are used. Moreover, we see that Table 16 demonstrates that, on average, learning a per-class shift increases performance by around 0.5% on average over different prototype settings. This demonstrates that learning per-class shifts allows the model to capture both dataset-level and class-level distribution shifts in a domain gap.

D.6. Prototype Variants

We explore different methods for creating class prototypes. Specifically, we experiment with different forms of aggregating the text encoded with the 80 ImageNet context prompts from CLIP [35] and our LLM-generated descrip-

Method	Flower102	DTD	Pets	Cars	UCF101	CalTech101	Food101	SUN397	Aircraft	EuroSAT	Average
TPT [38]	68.79 (±.1)	46.79 (±.1)	87.09 (±.1)	66.38 (±.2)	67.86 (±.1)	94.13 (±.1)	84.67 (±.1)	65.41 (±.1)	23.44 (±.3)	42.78 (±.3)	64.73 (±.1)
TPT + (templates + descriptors)*	69.67 (±.11)	47.56 (±.55)	87.88 (±.02)	66.91 (±.17)	68.35 (±.21)	94.17 (±.13)	84.89 (±.07)	66.23 (±.12)	23.55 (±.31)	43.12 (±.18)	65.23 (±.06)
Ours	**71.47** (±.12)	**51.00** (±.47)	**87.45** (±.09)	**68.99** (±.10)	**70.98** (±.24)	**94.90** (±.16)	**85.15** (±.08)	**68.85** (±.16)	**25.82** (±.45)	**44.61** (±.11)	**66.92** (±.04)

Table 8. Acc@1 of zero-shot image classification with CLIP-ViT-B/16 backbone on cross-dataset generalization over 3 random seeds. Best performances are in **bold**.

Method	ImageNet (IN)	IN-OOD Avg	Cross-Dataset Avg
Scale	70.40	61.12	65.96
Shift	<u>71.45</u>	<u>64.15</u>	**66.96**
Scale & Shift	**71.47**	**64.16**	<u>66.91</u>
FiLM	0.09	0.28	2.08

Table 9. Acc@1 of zero-shot image classification comparing different learned feature transformations using CLIP-ViT-B/16 backbone. Best performances are in **bold**. Second best performances are <u>underlined</u>.

# Steps	ImageNet (IN)	IN-OOD Avg	Cross-Dataset Avg
1	71.45	64.15	66.96
2	71.51	64.18	66.88
4	71.36	64.13	66.81

Table 10. Acc@1 of zero-shot image classification with CLIP-ViT-B/16 backbone. Performance improvements over zero-shot CLIP are denoted in (↑blue). Best performances are in **bold**.

tors. The CLIP ImageNet templates are class-agnostic and add image-level characteristics whereas the descriptors are class-specific and add class-level semantic information.

Tables 17 and 18 compare three variants of pooling these CLIP templated embeddings and descriptor embeddings to obtain a single class prototype. Similarly to the conclusion of Sec 3.4, we observe that in general, the gains observed using more advanced prototypes in the zero-shot setting almost directly translate to the test-time adaptation setting with shifting. In Sec 3, we present the results of our method using prototypes that are a micro average of the CLIP templates and LLM-generated descriptors.

E. Research Impact and Limitations

We propose TPS, a framework that can be used to easily and effectively improve zero-shot generalization of VLMs. Given the large-scale training of foundation VLMs, we believe it is important to understand different ways to better leverage the resulting rich multi-modal contrastive representation spaces in parameter- and runtime-constrained settings. We propose to learn a slight perturbation to the class prototypes to maintain the overall representation quality of the pre-trained embedding space while learning a better alignment to the OOD target dataset. We hope that this framework can inspire future work to explore other tasks where learning directly in the feature space can be an efficient alternative to more complex tuning approaches.

Our work builds on the CLIP [35] representation space and uses GPT-4 [31] to generate class descriptors to create more advanced class prototypes. Thus, our model has the potential to magnify the biases of both these models. Future studies may explore how to best leverage these models' capabilities without promoting its biases.

Method	ImageNet	ImageNet-A	ImageNet-V2	ImageNet-R	ImageNet-Sketch	Average	OOD Average
Zero-Shot	76.28	71.03	70.46	87.68	59.65	73.02	72.20
Ours (TPS)	77.66	79.05	72.41	89.95	61.57	76.13	75.75
Δ	**+ 1.38**	**+ 8.02**	**+ 1.95**	**+ 2.27**	**+ 1.92**	**+ 3.11**	**+ 3.55**

Table 11. Acc@1 for zero-shot and TPS on ImageNet and its out-of-distribution variants using CLIP-ViT-L/14 backbone.

Method	Flower102	DTD	Pets	Cars	UCF101	CalTech101	Food101	SUN397	Aircraft	EuroSAT	Average
Zero-Shot	78.52	59.57	93.95	76.87	77.00	96.51	89.63	71.66	30.93	62.52	73.72
Ours (TPS)	77.67	60.40	93.65	78.76	78.38	96.84	89.82	72.39	33.03	60.67	74.16
Δ	- 0.85	**+ 0.83**	- 0.30	**+ 1.89**	**+ 1.38**	**+ 0.33**	**+ 0.19**	**+ 0.73**	**+ 2.10**	- 1.85	**+ 0.44**

Table 12. Acc@1 for zero-shot and TPS on cross-domain generalization datasets using CLIP-ViT-L/14 backbone.

Prompt Type	Setting	ImageNet	ImageNet-A	ImageNet-V2	ImageNet-R	ImageNet-Sketch	Average	OOD Average
Vanilla	Zero-Shot	66.74	47.79	60.89	73.99	46.12	59.10	57.20
	+ shift	68.81 (±.03)	58.11 (±.16)	63.51 (±.17)	76.98 (±.05)	48.11 (±.09)	63.10 (±.08)	61.68 (±.09)
	Δ	**+ 2.07**	**+ 10.32**	**+ 2.62**	**+ 2.99**	**+ 1.99**	**+ 4.00**	**+ 4.48**
CoOp [49]	Zero-Shot	71.51	49.71	64.20	75.21	47.99	61.72	59.28
	+ shift	73.76 (±.04)	60.43 (±.12)	66.84 (±.10)	77.39 (±.05)	49.08 (±.06)	65.50 (±.02)	63.44 (±.03)
	Δ	**+ 2.25**	**+ 10.72**	**+ 2.64**	**+ 2.18**	**+ 1.09**	**+ 3.78**	**+ 4.16**
CLIP templates	Zero-Shot	68.35	49.95	61.97	77.59	48.21	61.21	59.43
	+ shift	70.39 (±.06)	60.47 (±.07)	64.66 (±.04)	80.70 (±.04)	50.38 (±.14)	65.32 (±.03)	64.05 (±.02)
	Δ	**+ 2.04**	**+ 10.52**	**+ 2.69**	**+ 3.11**	**+ 2.17**	**+ 4.11**	**+ 4.62**
Descriptors	Zero-Shot	68.52	48.91	61.78	74.81	47.68	60.34	58.29
	+ shift	70.38 (±.03)	59.21 (±.09)	63.80 (±.07)	77.49 (±.12)	49.57 (±.06)	64.09 (±.02)	62.52 (±.03)
	Δ	**+ 1.86**	**+ 10.30**	**+ 2.02**	**+ 2.68**	**+ 1.89**	**+ 3.75**	**+ 4.23**
CLIP templates	Zero-Shot	69.54	50.51	63.01	77.18	48.84	61.82	59.88
+ Descriptors	+ shift	71.43 (±.06)	60.78 (±.21)	65.00 (±.09)	80.06 (±.13)	50.97 (±.09)	65.65 (±.06)	64.20 (±.08)
	Δ	**+ 1.89**	**+ 10.27**	**+ 1.99**	**+ 2.88**	**+ 2.13**	**+ 3.83**	**+ 4.32**

Table 13. Acc@1 for zero-shot and with feature-space shift with features initialized using different prototype generation techniques on ImageNet and its out-of-distribution variants. Results are over 3 random seeds.

Prompt Type	Setting	Flower102	DTD	Pets	Cars	UCF101	CalTech101	Food101	SUN397	Aircraft	EuroSAT	Average
Vanilla	Zero-Shot	67.28	44.44	87.98	65.24	65.08	92.98	83.80	62.55	23.70	41.42	63.45
	+ shift	67.75 (±.10)	45.69 (±.10)	87.57 (±.10)	67.60 (±.23)	66.79 (±.21)	93.79 (±.08)	84.62 (±.03)	64.58 (±.03)	24.75 (±.39)	41.35 (±.03)	64.45 (±.04)
	Δ	**+ 0.47**	**+ 1.25**	- 0.41	**+ 2.36**	**+ 1.71**	**+ 0.81**	**+ 0.82**	**+ 2.03**	**+ 1.05**	- 0.07	**+ 1.00**
CLIP templates	Zero-Shot	65.57	44.86	88.25	66.19	67.46	93.67	83.77	65.78	23.64	47.74	64.69
	+ shift	66.41 (±.05)	45.61 (±.19)	87.99 (±.10)	68.66 (±.31)	68.02 (±.11)	93.85 (±.14)	84.54 (±.08)	67.19 (±.05)	24.66 (±.13)	48.28 (±.20)	65.52 (±.05)
	Δ	**+ 0.84**	**+ 0.75**	- 0.26	**+ 2.47**	**+ 0.56**	**+ 0.18**	**+ 0.77**	**+ 1.41**	**+ 1.02**	**+ 0.54**	**+ 0.83**
Descriptors	Zero-Shot	71.13	52.72	86.75	65.15	70.53	94.08	84.12	67.10	25.26	43.31	66.02
	+ shift	71.69 (±.15)	53.80 (±.21)	87.82 (±.19)	67.00 (±.14)	71.18 (±.15)	94.56 (±.08)	84.78 (±.05)	68.25 (±.18)	26.27 (±.09)	42.11 (±.18)	66.75 (±.06)
	Δ	**+ 0.56**	**+ 1.08**	**+ 1.07**	**+ 1.85**	**+ 0.65**	**+ 0.48**	**+ 0.66**	**+ 1.15**	**+ 1.01**	- 1.20	**+ 0.73**
CLIP templates	Zero-Shot	70.52	49.94	87.22	66.48	70.24	94.12	84.47	67.55	24.69	44.14	65.94
+ Descriptors	+ shift	71.47 (±.12)	51.00 (±.47)	87.45 (±.09)	68.99 (±.10)	70.98 (±.24)	94.90 (±.16)	85.15 (±.08)	68.85 (±.16)	25.82 (±.45)	44.61 (±.11)	66.92 (±.04)
	Δ	**+ 0.95**	**+ 1.06**	**+ 0.23**	**+ 2.51**	**+ 0.74**	**+ 0.78**	**+ 0.68**	**+ 1.30**	**+ 1.13**	**+ 0.47**	**+ 0.98**

Table 14. Acc@1 for zero-shot and with feature-space shift with features initialized using different prototype generation techniques on cross-domain generalization datasets. Results are over 3 random seeds.

Method	ImageNet	ImageNet-A	ImageNet-V2	ImageNet-R	ImageNet-Sketch	Average	OOD Average
Shared	71.23 (±.02)	56.57 (±.19)	64.98 (±.03)	79.31 (±.03)	50.80 (±.06)	64.58 (±.04)	62.92 (±.06)
Per class	**71.43** (±.06)	**60.78** (±.21)	**65.00** (±.09)	**80.06** (±.13)	**50.97** (±.09)	**65.65** (±.06)	**64.20** (±.08)

Table 15. Acc@1 for learning a shared vs. per-class shift on top of different prototypes over 3 random seeds. Best performances are in **bold**.

Method	Flower102	DTD	Pets	Cars	UCF101	CalTech101	Food101	SUN397	Aircraft	EuroSAT	Average
Shared	71.36 (±.12)	50.49 (±.12)	**87.46** (±.12)	67.33 (±.06)	70.77 (±.12)	94.35 (±.06)	84.82 (±.01)	68.12 (±.04)	25.27 (±.02)	**44.67** (±.06)	66.47 (±.03)
Per-class	**71.47** (±.12)	**51.00** (±.47)	87.45 (±.09)	**68.99** (±.10)	**70.98** (±.24)	**94.90** (±.16)	**85.15** (±.08)	**68.85** (±.16)	**25.82** (±.45)	44.61 (±.11)	**66.92** (±.04)

Table 16. Acc@1 for learning a shared vs. per-class shift on top of different prototypes over 3 random seeds. Best performances are in **bold**.

Prompt Type(s)	Pooling Method	ImageNet	ImageNet-A	ImageNet-V2	ImageNet-R	ImageNet-Sketch	Average	OOD Average
					Zero-Shot			
Vanilla prompt	N/A	66.74	47.79	60.89	73.99	46.12	59.10	57.20
CLIP templates + Descriptors	Macro	68.73	50.32	62.31	**77.67**	48.56	61.52	59.72
CLIP templates + Descriptors	Micro	**69.54**	50.51	**63.01**	77.18	48.84	**61.82**	**59.88**
CLIP templates × Descriptors	Macro	69.03	**50.73**	62.22	76.91	**49.07**	61.59	59.73
					With Shift			
Vanilla prompt	N/A	68.81 (±.03)	58.11 (±.16)	63.51 (±.17)	76.98 (±.05)	48.11 (±.09)	63.10 (±.08)	61.68 (±.09)
CLIP templates + Descriptors	Macro	70.75 (±.08)	**60.86** (±.09)	64.95 (±.11)	**80.84** (±.03)	50.70 (±.11)	65.62 (±.02)	**64.34** (±.02)
CLIP templates + Descriptors	Micro	**71.43** (±.06)	60.78 (±.21)	**65.00** (±.09)	80.06 (±.13)	50.97 (±.09)	**65.65** (±.06)	64.20 (±.08)
CLIP templates × Descriptors	Macro	70.82 (±.02)	60.42 (±.06)	64.50 (±.05)	79.53 (±.09)	**51.13** (±.02)	65.28 (±.01)	63.89 (±.02)

Table 17. Acc@1 for different variants of prototype generation, i.e. ways of combining templates and descriptors, on natural distribution shifts, over 3 random seeds. Best performances for each setting are in **bold**.

Prompt Type(s)	Pooling Method	Flower102	DTD	Pets	Cars	UCF101	CalTech101	Food101	SUN397	Aircraft	EuroSAT	Average
							Zero-Shot					
Vanilla prompt	N/A	67.28	44.44	87.98	65.24	65.08	92.98	83.80	62.55	23.70	41.42	63.45
CLIP templates + Descriptors	Macro	66.91	45.86	**88.33**	66.46	68.12	93.83	83.97	66.34	24.03	46.62	65.05
CLIP templates + Descriptors	Micro	70.52	49.94	87.22	**66.48**	70.24	94.12	**84.47**	67.55	24.69	44.14	65.94
CLIP templates × Descriptors	Macro	**72.03**	**50.83**	86.21	66.12	**70.90**	**94.16**	83.73	**67.98**	**25.53**	**47.19**	**66.47**
							With Shift					
Vanilla prompt	N/A	67.75 (±.10)	45.69 (±.10)	87.57 (±.10)	67.60 (±.23)	66.79 (±.21)	93.79 (±.08)	84.62 (±.03)	64.58 (±.03)	24.75 (±.39)	41.35 (±.03)	64.45 (±.04)
CLIP templates + Descriptors	Macro	67.52 (±.27)	46.43 (±.28)	**88.00** (±.13)	**69.04** (±.16)	68.67 (±.18)	94.16 (±.18)	84.77 (±.04)	67.70 (±.08)	24.79 (±.30)	**47.09** (±.19)	65.82 (±.06)
CLIP templates + Descriptors	Micro	71.47 (±.12)	51.00 (±.47)	87.45 (±.09)	68.99 (±.10)	70.98 (±.24)	**94.90** (±.16)	**85.15** (±.08)	68.85 (±.16)	25.82 (±.45)	44.61 (±.11)	66.92 (±.04)
CLIP templates × Descriptors	Macro	**72.53** (±.12)	**52.56** (±.09)	86.15 (±.05)	68.89 (±.07)	**71.44** (±.20)	94.43 (±.06)	84.44 (±.08)	**69.04** (±.02)	**26.51** (±.26)	45.65 (±.15)	**67.16** (±.03)

Table 18. Acc@1 for different variants of knowledge injection, i.e. ways of combining templates and descriptors, over 3 random seeds on cross-dataset generalization tasks. Best performances in each setting are in **bold**.