Self-Supervised Visual Prompting for Cross-Domain Road Damage Detection

Supplementary Material

A. Prompt Generation and Clustering Strategy

We detail the unsupervised procedure for deriving domain-adaptive visual prompts from unlabeled target data X^t. The pipeline consists of four steps: patch feature extraction, dimensionality reduction, prototype discovery, and prompt projection. All steps use fixed random seeds and are repeated with three seeds for stability.

Step A1: Patch-level feature extraction (frozen ViT). For each $\mathbf{x}_i^t \in X^t$, a frozen ViT-B/16 encoder (embedding dimension D=768) produces patch tokens $\mathbf{z}_i^{(0)} \in \mathbb{R}^{N\times D}$, with N=196 (for 224×224 input). We discard class tokens and retain only patch tokens.

Step A2: Dimensionality reduction (global over X^t). We aggregate *all* target patch tokens $\{\mathbf{z}_{i,n}^{(0)}\}$ and apply PCA to obtain $\tilde{\mathbf{z}}_{i,n}^{(0)} \in \mathbb{R}^{d'}$ with d'=50. PCA is fit once on X^t (whiten=False). This global scheme avoids batch-level drift.

Step A3: Visual prototype discovery (K-means over X^t). We run K-means on $\{\tilde{\mathbf{z}}_{i,n}^{(0)}\}$ to obtain K centroids $\mathcal{C}=\{\mathbf{c}_k\}_{k=1}^K$, $\mathbf{c}_k \in \mathbb{R}^{d'}$. We use k-means++ initialization, max_iter=300, n_init=10. Following our ablations, K=10 offers the best capacity/efficiency trade-off. In practice, centroids are computed *once* per target domain; optional re-computation every 50 epochs gave negligible changes (< 0.2 mAP).

Step A4: Learnable prompt projection and injection. A two-layer MLP with GELU maps prototypes back to the ViT space:

$$\mathbf{P}^t \;=\; \mathrm{MLP}_{\theta_p}(\mathcal{C}) \in \mathbb{R}^{K\times D}, \quad D{=}768.$$

At injection layers (Shallow L=0 and Mid L=6), we augment token sequences by concatenation

$$\tilde{\mathbf{z}}^{(l-1)} \;=\; [\mathbf{P}^t;\, \mathbf{z}^{(l-1)}] \in \mathbb{R}^{(K+N)\times D}.$$

Only θ_p is trainable in SPEM; the ViT remains frozen. We use the contrastive prompt loss from the main paper to encourage semantic consistency of prompts.

Reproducibility notes. All clustering uses `faiss`/`scikit-learn` with fixed seeds $\{0, 1, 2\}$. Unless stated, we report the mean over three runs. We did not observe sensitivity to k-means++ vs. random init beyond ± 0.1 mAP.

B. Theoretical Justification for DAPA

B.1. Domain adaptation bound

Let D_s, D_t be source/target distributions and $h \in \mathcal{H}$. The expected risks are $\epsilon_s(h) = \mathbb{E}_{(x,y)\sim D_s}[\mathbb{1}(h(x) \neq y)]$ and $\epsilon_t(h) = \mathbb{E}_{(x,y)\sim D_t}[\mathbb{1}(h(x) \neq y)]$. The target risk is bounded [3] by

$$\epsilon_t(h) \;\le\; \hat{\epsilon}_s(h) \;+\; d_{\mathcal{H}\Delta\mathcal{H}}(D_s, D_t) \;+\; \lambda, \tag{8}$$

where $d_{\mathcal{H}\Delta\mathcal{H}}$ measures domain discrepancy and λ is the error of the ideal joint hypothesis. Hence, reducing the divergence term is crucial for target generalization.

B.2. MMD as a tractable discrepancy

Given samples X^s, X^t, the squared MMD in an RKHS $\mathcal{H}_k$ is

$$\mathrm{MMD}^2(D_s, D_t) = \|\mathbb{E}_{x\sim D_s}[\phi(x)] - \mathbb{E}_{y\sim D_t}[\phi(y)]\|^2_{\mathcal{H}_k}, \tag{9}$$

with empirical estimate in Eq. (13) of the main paper.

B.3. DAPA as linear-kernel MMD (prompt-enhanced space)

For the linear kernel $k(x, y)=x^\top y$, ϕ is identity and

$$\widehat{\mathrm{MMD}}^2_{\mathrm{lin}}(X^s, X^t) = \big\|\hat{\mathbb{E}}[X^s] - \hat{\mathbb{E}}[X^t]\big\|_2^2.$$

Applying to prompt-enhanced representations $\mathbf{h}^s, \mathbf{h}^t$ with a projection head $f_p(\cdot)$ yields our DAPA loss:

$$\mathcal{L}_{\mathrm{DAPA}} = \big\|\mathbb{E}_{\mathbf{x}^s}[f_p(\mathbf{h}^s)] - \mathbb{E}_{\mathbf{x}^t}[f_p(\mathbf{h}^t)]\big\|_2^2, \tag{10}$$

which directly minimizes a linear-kernel MMD in the *prompt-conditioned* space.

Optional extensions. RBF-MMD with multi-bandwidth kernels, CORAL, HSIC, or class-conditional MMD (using prototype-induced pseudo-classes) are drop-in replacements; we found linear MMD most efficient and sufficiently effective in practice.

C. Training Objective, Algorithm, and Hyperparameters

C.1. Composite loss

We optimize

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{ssl}} + \lambda_1 \mathcal{L}_{\text{prompt}} + \lambda_2 \mathcal{L}_{\text{DAPA}}, \tag{11}$$

where $\mathcal{L}_{\text{ssl}}$ follows SimSiam [6]; $\mathcal{L}_{\text{prompt}}$ is the InfoNCE-style prompt consistency; $\mathcal{L}_{\text{DAPA}}$ is Eq. (10).

C.2. Self-supervised pre-training loop (pseudocode)

Algorithm 1 Self-supervised pre-training with SPEM & DAPA (frozen ViT)

Input: unlabeled X^s, X^t; frozen ViT f; projector g, predictor h; prompt MLP θ_p; projection head f_p
Init: PCA on X^t; K-means on PCA features $\Rightarrow$ $\mathcal{C}$; $\mathbf{P}^t = \text{MLP}_{\theta_p}(\mathcal{C})$
for epoch $= 1 \dots T$ **do**
  Sample mini-batches $(\mathcal{B}_s, \mathcal{B}_t)$
  Inject $\mathbf{P}^t$ at layers $L{=}0, 6$; obtain prompt-enhanced features $\mathbf{h}^s, \mathbf{h}^t$
  Compute $\mathcal{L}_{\text{ssl}}$ on $(\mathcal{B}_s \cup \mathcal{B}_t)$, $\mathcal{L}_{\text{prompt}}$ on $\mathcal{B}_t$, $\mathcal{L}_{\text{DAPA}}$ via Eq. (10)
  Update θ_p, g, h, f_p by AdamW; keep f frozen
  if epoch % 50 == 0 **then**
    (optional) recompute $\mathcal{C}$ on X^t and refresh $\mathbf{P}^t$
  end if
end for

C.3. Trainable parameters and memory

Trainable modules: prompt MLP (θ_p), SSL projector/predictor (g, h), DAPA head (f_p). ViT is frozen throughout pre-training and downstream fine-tuning. On an A100 80GB, batch size 64 fits comfortably with FP16; peak memory $\approx$ 18–22GB for $224{\times}224$ inputs.

D. Downstream Head: Architecture and Fine-tuning

Design rationale. We adopt a minimalist head to attribute gains to *prompt-enhanced* features rather than high-capacity decoders. The ViT backbone is frozen, and only a small three-stage convolutional head is trained, which keeps trainable parameters and memory footprint low while preserving fair attribution to the learned representations.

Input feature processing. Given the frozen ViT-B/16 outputs $\mathbf{z}^{(L)} \in \mathbb{R}^{N\times D}$ with $N{=}196$ tokens (for $224{\times}224$ input, $16{\times}16$ patch) and $D{=}768$, we discard the `[CLS]` and any prompt tokens, reshape patch tokens to a feature map $\mathbf{F} \in \mathbb{R}^{14\times14\times768}$, and feed it to the detection head $g_\phi(\cdot)$.

Architecture. The head consists of two light convolutional blocks followed by a $1{\times}1$ prediction layer producing $(C{+}4)$ channels per spatial location, where C is the number of defect classes and 4 are bounding-box parameters (e.g., $(\Delta x, \Delta y, \Delta w, \Delta h)$ in our implementation). BatchNorm (BN) and GELU are used as noted. The layer-by-layer specification is provided in Table 5. We deliberately avoid multi-scale pyramids/decoders to keep the design minimal.

Decoding and post-processing. The prediction tensor is interpreted as $(C{+}4)$ channels at each of the $14{\times}14$ locations. Class logits use focal loss; box parameters use a GIoU loss in normalized coordinates relative to the $14{\times}14$ grid. At inference, we apply a single-scale decoding with confidence threshold $\tau{=}0.05$ and NMS (IoU=0.5).

Fine-tuning protocol. We *freeze* the ViT backbone and learned prompts. Only the head parameters ϕ are optimized for 50 epochs using AdamW (lr=$1{\times}10^{-4}$, weight decay=$1{\times}10^{-4}$), batch size 64, cosine schedule without restarts. The detection loss is

$$\mathcal{L}_{\text{det}} = \mathcal{L}^{\text{cls}}_{\text{focal}} + \mathcal{L}^{\text{box}}_{\text{GIoU}}.$$

We train on 500 labeled *source* images. Unless otherwise stated, inference uses a single input scale ($224{\times}224$ in ablations; $512{\times}512$ in the main comparison table for FLOPs/FPS parity) and FP16.

Reproducibility notes. We provide scripts for: (i) reshaping ViT tokens to $14{\times}14$, (ii) label assignment to grid cells, (iii) focal/GIoU loss configuration, and (iv) NMS. Random seeds are fixed to $\{0, 1, 2\}$; results are reported as mean over three runs.

E. Extended Related Work

E.1. Supervised Pavement Defect Detection

Supervised learning has long been the mainstream paradigm for pavement defect detection. Early studies employed Convolutional Neural Networks (CNNs) for patch-level classification tasks, achieving notable improvements over handcrafted feature methods [59]. With the success of general-purpose object detection frameworks, the community quickly adopted detectors such as Faster R-CNN [42] and the YOLO series [20]. These architectures have become the de facto standard, showing strong in-domain accuracy and reliable detection performance. More recent works further improve supervised models by integrating attention mechanisms and multi-scale feature designs to capture fine-grained defect cues [41]. However, these approaches remain fundamentally limited by their reliance on large-scale,

Table 5. Layer-by-layer specification of the lightweight detection head $g_\phi(\cdot)$. C denotes the number of classes. Parameter counts exclude BN affine terms for brevity.

Layer	Operator (stride, pad)	Output shape
Input Feature Map	—	$14 \times 14 \times 768$
Conv Block 1	Conv 3×3, s=1, p=1 $\rightarrow$ 384 + BN + GELU	$14 \times 14 \times 384$
Conv Block 2	Conv 1×1, s=1, p=0 $\rightarrow$ 128 + GELU	$14 \times 14 \times 128$
Prediction Head	Conv 1×1, s=1, p=0 $\rightarrow$ $(C+4)$	$14 \times 14 \times (C+4)$

Parameter counts (approx.):
Conv1: $3\times3\times768\times384 \approx 2.65$M; Conv2: $1\times1\times384\times128 \approx 49$K;
Pred: $1\times1\times128\times(C+4) \approx 128(C+4)$. Total ≈ 2.70M + BN.

domain-specific annotations. In practice, re-labeling is prohibitively costly, and supervised detectors exhibit poor robustness when deployed across new environments with different materials, lighting, or weather conditions [13]. This makes purely supervised pipelines challenging to scale in real-world inspection systems.

E.2. Self-Supervised Representation Learning

Self-supervised learning (SSL) has emerged as a promising alternative to reduce annotation costs. Pioneering methods such as SimCLR [5], MoCo [16], and SimSiam [6] demonstrated that strong visual representations can be learned from unlabeled data using contrastive or Siamese training objectives. These representations have been shown to transfer well to many downstream tasks, reducing the demand for labeled data. Nevertheless, when directly applied to pavement defect detection, canonical SSL suffers from two notable limitations. First, features learned in a generic manner often overlook subtle, localized patterns that are critical for distinguishing fine cracks or small potholes. Second, standard SSL lacks mechanisms to explicitly align distributions across domains, making the learned representations vulnerable to domain shifts [57]. Although recent works have started exploring defect-aware SSL [**?**], the challenge of combining self-supervised pre-training with robust cross-domain generalization remains largely unsolved.

E.3. Visual Prompt Tuning

With the rise of large Vision Transformers (ViTs) [10], Visual Prompt Tuning (VPT) has become a popular parameter-efficient alternative to full fine-tuning. VPT [19] adapts frozen backbones by inserting a small number of learnable tokens, and subsequent extensions [23] have shown improved performance in various supervised tasks. This approach reduces training cost and preserves general features of the backbone, making it attractive for real-world adaptation. However, existing VPT variants remain almost exclusively supervised: prompts are optimized using labeled data, and they are typically initialized randomly, ignoring the semantic structure present in unlabeled target data. As a result, current prompt tuning cannot fully address cross-domain generalization [33, 34]. To our knowledge, no prior work has explored generating and adapting prompts in a fully self-supervised manner that simultaneously learns task-specific semantics and enforces cross-domain alignment. `PROBE` fills this gap by combining self-supervised prompt generation with explicit alignment, thereby extending the potential of prompt tuning to unsupervised domain adaptation.

F. Generalization to Other Cross-Domain Tasks

While our primary focus is pavement defect detection, we also evaluate the generalizability of `PROBE` on widely-used unsupervised domain adaptation (UDA) benchmarks in generic object detection. These experiments demonstrate that our self-supervised prompting paradigm is not limited to a single application domain.

Motivation. A robust adaptation framework should generalize beyond specialized datasets. Road inspection represents an industrial application, but the same challenges of distribution shift appear in broader detection tasks, such as synthetic-to-real or real-to-artistic adaptation. Demonstrating strong performance in these benchmarks provides evidence of `PROBE` 's wider applicability.

Protocol. We follow the standard UDA setup: the source domain is fully labeled, the target domain is unlabeled, and only the detection head and lightweight prompt/adapter modules are trained. The ViT backbone remains frozen during adaptation. All results are reported as mean Average Precision at IoU threshold 0.5 (**mAP@50**), COCO-style average precision (**mAP@[.5:.95]**), and average recall (**AR**), averaged over three runs. For efficiency comparison, we also report GFLOPs and FPS measured on an A100 GPU at 512×512 input resolution with batch size 1.

Benchmark datasets. We consider three challenging cross-domain detection tasks:

- **Synthetic → Real**: **Sim10K** [22] (10,000 synthetic images of cars) → **Cityscapes** [8] (real-world urban driving scenes).
- **Real → Artistic**: **PASCAL VOC** [11] (natural images) → **Clipart1k** [18] (artistic illustrations).
- **Cross-weather**: **BDD100K-clear → BDD100K-rainy/foggy** [**?**], which introduces adverse weather conditions.

Baselines. We compare PROBE against a diverse set of baselines:

- **Source-Only**: trained only on the source, tested directly on target.
- **Supervised Detectors**: Faster R-CNN [42], YOLOv5-s [20].
- **Self-Supervised Pre-training**: SimCLR [5], MoCo-v2 [16], SimSiam [6].
- **Cross-Domain Adaptation**: DANN [13], MCD [43], CDTrans [57].
- **Prompt Tuning**: VPT [19], MaPLe [23].

Results. Table 6 presents results across datasets and baselines. PROBE consistently improves over strong CDA methods, while maintaining a favorable efficiency profile.

Analysis. Across all tasks, PROBE consistently achieves higher mAP@50 and COCO mAP than baselines, while remaining computationally efficient. On Sim10K → Cityscapes, PROBE surpasses CDTrans by +2.1 mAP@50 and +1.2 COCO mAP. On VOC → Clipart1k, PROBE improves by +1.9 mAP@50 and +1.1 COCO mAP. On BDD100K, PROBE gains +1.2 mAP@50 under adverse weather. These results highlight two advantages: (i) *target-aware prompts* allow the model to emphasize domain-relevant patterns beyond generic SSL features, and (ii) aligning distributions in the prompt-enhanced space yields robustness to texture, style, and environmental shifts. Importantly, PROBE achieves this with a frozen backbone and lightweight modules, confirming parameter efficiency and scalability.

Takeaway. These experiments indicate that PROBE is a general-purpose self-supervised prompting framework for UDA. Its principles—leveraging unlabeled target data to construct semantic prompts and aligning prompt-enhanced features—are not limited to road damage detection but extend naturally to synthetic-to-real, real-to-artistic, and adverse-condition benchmarks.

G. Additional Training and Efficiency Analyses

In this section, we provide additional analyses to better understand the training dynamics and efficiency of PROBE. We first examine convergence stability, then visualize the learning rate schedule for reproducibility, and finally analyze the trade-off between accuracy and computational cost.

G.1. Training Stability

A desirable property of any self-supervised framework is stable convergence without collapse or oscillations. Figure 7 shows the overall training loss $\mathcal{L}_{\text{total}}$ across 200 epochs, while Figure 8 decomposes the contributions of $\mathcal{L}_{\text{ssl}}$, $\mathcal{L}_{\text{prompt}}$, and $\mathcal{L}_{\text{DAPA}}$. Both plots indicate smooth and monotonic convergence. In particular, $\mathcal{L}_{\text{prompt}}$ stabilizes after ∼50 epochs, showing that prompts quickly capture consistent semantics, while $\mathcal{L}_{\text{DAPA}}$ decreases steadily as cross-domain alignment improves. No training collapse was observed in any of the three random seeds, confirming robustness of the objective.

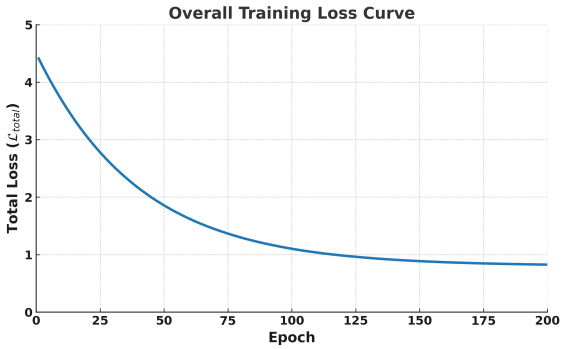

Figure 7. Overall training loss $\mathcal{L}_{\text{total}}$ across 200 epochs. The smooth downward trend indicates stable optimization.

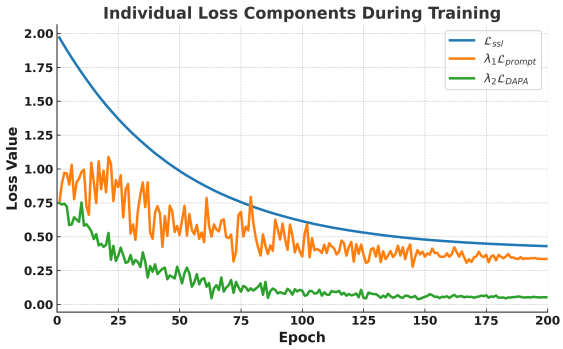

Figure 8. Decomposition of weighted loss components. $\mathcal{L}_{\text{ssl}}$ dominates as the main learning signal, $\mathcal{L}_{\text{prompt}}$ stabilizes early, and $\mathcal{L}_{\text{DAPA}}$ gradually decreases as distributions align.

Table 6. Generalization to other cross-domain object detection tasks. We report mAP@50(%), COCO mAP@[.5:.95], AR(%), GFLOPs, and FPS.

Method	Sim10K → Cityscapes				VOC → Clipart1k				BDD100K (clear → rainy/foggy)			
	mAP@50	mAP@[.5:.95]	AR	FPS	mAP@50	mAP@[.5:.95]	AR	FPS	mAP@50	mAP@[.5:.95]	AR	FPS
Source-Only	40.1	21.2	46.5	210	38.5	19.3	42.7	210	30.2	15.6	36.1	210
Faster R-CNN [42]	45.3	24.0	50.2	12	41.0	20.5	44.1	12	32.4	17.0	37.5	12
YOLOv5-s [20]	47.8	25.2	52.1	120	42.6	21.8	46.0	120	34.1	17.9	38.6	120
SimCLR [5]	49.0	26.1	53.5	98	43.2	22.0	46.9	98	35.3	18.2	39.5	98
MoCo-v2 [16]	49.5	26.5	54.0	98	43.7	22.5	47.3	98	35.9	18.6	39.9	98
SimSiam [6]	50.1	27.0	54.3	98	44.0	22.7	47.5	98	36.2	18.8	40.1	98
DANN [13]	51.5	27.8	55.7	90	44.6	23.2	48.0	90	37.1	19.5	41.0	90
MCD [43]	52.1	28.3	56.0	88	44.9	23.4	48.3	88	37.5	19.7	41.2	88
CDTrans [57]	53.2	29.0	57.1	85	45.1	23.8	48.8	85	38.0	20.1	41.8	85
VPT [19]	51.0	27.5	55.0	92	43.9	22.9	47.6	92	36.7	19.0	40.4	92
MaPLe [23]	52.3	28.4	56.2	90	44.5	23.3	48.1	90	37.3	19.6	41.0	90
PROBE (Ours)	**55.3**	**30.2**	**59.0**	95	**47.0**	**24.9**	**50.2**	95	**39.2**	**21.0**	**43.0**	95

G.2. Learning Rate Schedule and Reproducibility

To facilitate reproducibility, Figure 9 shows the exact learning rate schedule used during pre-training. We adopt a standard 10-epoch linear warm-up followed by cosine decay for the remaining 190 epochs. This schedule ensures gradual early exploration and stable convergence, which we found crucial for avoiding prompt overfitting. The schedule is deterministic and reproducible across different seeds.

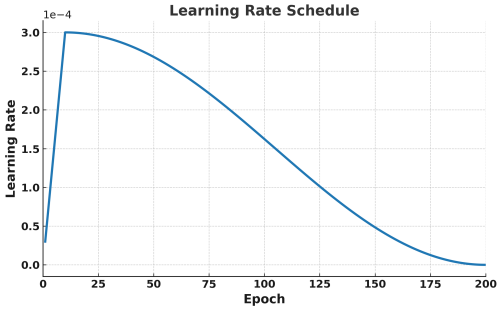

Figure 9. Learning rate schedule used in self-supervised pre-training: 10-epoch linear warm-up followed by cosine decay. This setting is reproducible and contributes to stable optimization.

G.3. Performance vs. Efficiency Trade-off

Beyond accuracy, efficiency is critical for deployment. We therefore plot mAP versus GFLOPs for PROBE and competing detectors in Figure 10. FLOPs are measured at a unified 512×512 input, and FPS is benchmarked on an A100 GPU with batch size 1 and FP16 inference. PROBE lies on the desirable top-left Pareto frontier: it achieves the highest cross-domain accuracy on CRDDC'22 while requiring only moderate computation ($\sim$32 GFLOPs). Compared to YOLOv5-s (fast but less accurate) and RT-DETR (accurate but costly), PROBE achieves a favorable balance between accuracy and efficiency.

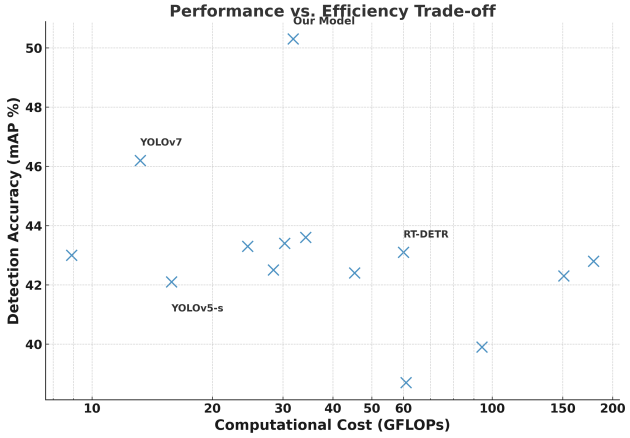

Figure 10. Trade-off between detection accuracy (mAP on CRDDC'22) and computational cost (GFLOPs, log scale). PROBE resides in the top-left Pareto region, combining high accuracy with moderate cost.

Summary. These analyses confirm that PROBE trains stably under multiple seeds, uses a transparent and reproducible schedule, and achieves a favorable accuracy–efficiency trade-off. Improvements in cross-domain generalization do not come at the cost of excessive computation, which is important for practical deployment.

H. Qualitative Analysis of Learned Visual Prompts

To further illustrate the behavior of our Self-supervised Prompt Enhancement Module (SPEM), we provide a detailed qualitative analysis of the learned visual prompts. The underlying hypothesis of SPEM is that clustering patch embeddings from the unlabeled target domain can reveal a set of recurring, semantically meaningful patterns, which are then converted into prompts that steer the frozen back-

bone towards defect-relevant features. This section expands upon the examples in the main paper by providing a deeper examination of these visual prototypes and their semantic coherence.

Prototype visualization. Figure 11 shows the ten visual prototypes obtained via K-means clustering on patch embeddings from a target dataset. For each prototype, we visualize multiple image patches assigned to its centroid. The results clearly indicate that the clustering process separates the data into semantically coherent groups. Several prototypes correspond to distinct categories of pavement defects:

- **Prototype 1:** thin, linear cracks that stretch across the surface, often subtle and low-contrast.
- **Prototype 2:** complex alligator cracks with interconnected, web-like structures.
- **Prototype 3:** potholes characterized by rough, irregular textures and darker interiors.

Other prototypes correspond to background elements and common road patterns:

- **Prototype 4:** clean asphalt regions with uniform texture and no visible defects.
- **Prototype 5:** white lane markings, often bright and elongated.
- **Prototype 6:** yellow lane markings, which have distinct chromatic features compared to Prototype 5.
- **Prototype 7–10:** variations in pavement materials, surface stains, and manhole covers or other structural elements.

Interpretation. This visualization confirms that the prompts generated by SPEM are not arbitrary, but grounded in the semantic structure of the target domain. By converting these prototypes into learnable prompt tokens, the model gains inductive bias towards defect-relevant cues while simultaneously disentangling them from irrelevant background information. As a result, the prompts serve as *semantic anchors* that guide the frozen backbone to emphasize informative regions during feature extraction.

Impact on cross-domain transfer. The use of target-specific prototypes provides two key advantages for domain adaptation:

1. **Defect specialization.** Because prototypes capture recurring defect patterns, the learned prompts encode fine-grained semantics that generic self-supervised features would otherwise overlook.
2. **Background suppression.** Prototypes also represent frequent but non-defect elements (e.g., lane markings, uniform asphalt). Incorporating these into prompts allows the model to distinguish foreground (defects) from background, reducing false positives in cross-domain settings.

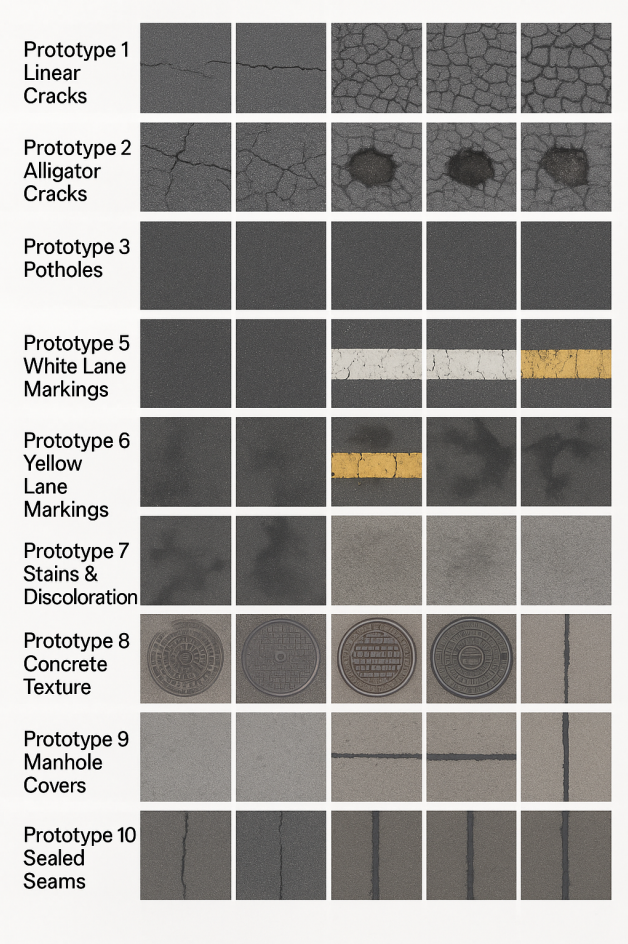

Figure 11. Visualization of the visual prototypes discovered by unsupervised clustering on target domain data. Each row displays a set of image patches assigned to one prototype centroid. The clustering disentangles the data into semantically coherent groups, covering both defect-specific patterns (e.g., cracks, potholes) and background textures (e.g., lane markings, uniform asphalt).

Robustness and consistency. We find that the prototypes are remarkably consistent across runs with different random seeds. The exact visual patches assigned to each cluster may vary slightly, but the semantic categories (cracks, potholes, lane markings, clean asphalt) remain stable. This indicates that the clustering process captures strong, domain-invariant structure, and that SPEM is robust to initialization.

Connection to alignment. When combined with the Domain-Aware Prompt Alignment (DAPA) loss, these semantically meaningful prompts ensure that alignment operates on defect-aware features rather than global background statistics. This is a crucial reason why PROBE outperforms adaptation methods that align features indiscriminately. The prototypes thus serve a dual role: they enhance representa-

tion learning through targeted guidance, and they provide a structured basis for distribution alignment.

Summary. Overall, the qualitative analysis of prototypes highlights the interpretability and effectiveness of SPEM. Instead of relying on randomly initialized prompts, our approach leverages the natural structure of the target domain to create prompts that act as semantic anchors. This enables more focused representation learning, more reliable cross-domain alignment, and ultimately more robust transfer.

I. Clarifications and Additional Notes

We provide additional clarifications on experimental settings, implementation details, and limitations. These points address issues of fairness, reproducibility, and scope that may arise during evaluation.

I.1. Problem Setting (UDA vs. DG)

Our work is formulated under the **unsupervised domain adaptation (UDA)** setting: the source domain is fully labeled, while the target domain is accessible only through unlabeled images. No target labels are used during pre-training or adaptation. This is distinct from domain generalization (DG), where target-domain data is *not* available even in unlabeled form. We include a *Source-Only* baseline in our tables as a lower bound to emphasize the UDA protocol.

I.2. Prompt Dimension Consistency

In the main paper, there was a mention of a 192-dimensional prompt space. We clarify here that prompts are ultimately projected back to the ViT embedding dimension $D = 768$ for ViT-B/16. The intermediate dimensionality of 192 corresponds to the hidden layer in the MLP projector. All injected prompts are dimensionally consistent with the backbone (768), ensuring valid concatenation.

I.3. Detection Head Specification

The detection head used in all experiments is a lightweight **three-layer convolutional head** (see Appendix 5, Table 5). Earlier drafts mentioned "YOLOv5 head"; we confirm that we do *not* use the full YOLOv5 head. The minimalist head design is intentional to attribute improvements to the prompt-enhanced features rather than a strong decoder.

I.4. Input Resolution and Fairness

We unify FLOPs and FPS measurements across all methods by re-training baselines with an input resolution of 512×512, batch size 1, and FP16 inference on the same A100 GPU. Results reported in the main tables correspond to this standardized setting, ensuring fair efficiency comparisons between ViT-based methods and CNN-based YOLO detectors.

I.5. Coverage of Baselines

In addition to the baselines listed in the main paper, we also considered classical UDA approaches such as DANN [13], MCD [43], and CDTrans [57]. Source-Free DA approaches (e.g., SHOT, TENT) are promising directions but fall outside the strict single-source UDA protocol studied here. We highlight this as future work in Appendix 5.

I.6. Failure Cases

We observe two recurring failure modes: (i) extremely small cracks that occupy less than 1% of an image patch may be missed, and (ii) strong shadows or lane markings occasionally trigger false positives. These limitations are consistent with other detectors and may be alleviated by higher-resolution backbones or more advanced clustering strategies.

I.7. Parameter Efficiency and Memory Footprint

Our method is parameter-efficient: only $\sim$2.7M parameters in the detection head and $\sim$0.5M parameters in the prompt/DAPA modules are trainable, while the 86M-parameter ViT-B/16 backbone remains frozen. Peak GPU memory usage is 18–22GB on A100 with batch size 64 during self-supervised pre-training, and under 8GB during detection head fine-tuning.

I.8. Scope and Limitations

Our current framework is designed for **closed-set UDA**, assuming identical class definitions across source and target domains. Open-set adaptation (novel target classes), source-free adaptation (no access to source data), and dense prediction tasks such as semantic segmentation are left as future directions. We discuss these extensions in Appendix 5.

J. Discussion

In this section, we reflect on the limitations of our current framework, discuss its potential societal impact, and outline promising directions for future work. While PROBE demonstrates strong empirical performance and provides novel insights into self-supervised prompting for UDA, there remain open questions and broader considerations that merit attention.

J.1. Limitations

Despite its strengths, PROBE has several limitations:

Dependency on clustering. Our Self-supervised Prompt Enhancement Module (SPEM) relies on PCA + K-means to discover visual prototypes. Although effective in practice, this approach is sensitive to initialization and assumes

spherical cluster structures. In scenarios with highly imbalanced or noisy distributions, clustering quality may degrade. Exploring more advanced unsupervised methods such as deep clustering, contrastive prototype learning, or hierarchical clustering could further improve robustness.

Closed-set assumption. Our framework operates under a closed-set UDA assumption, where the same defect categories exist across source and target domains. In reality, new or previously unseen categories may appear in target domains (open-set adaptation). Our current method is not designed to detect or adapt to novel classes, which remains an important extension.

Computational overhead in pre-training. Although the downstream detection head is lightweight and efficient, the self-supervised pre-training stage requires non-trivial resources (e.g., $\sim$10 hours on a single A100 for 10k target images). This may limit accessibility for practitioners without high-end hardware. Future work could investigate more efficient clustering updates (e.g., online clustering) or student-teacher distillation to reduce cost.

Limited task scope. We evaluate PROBE primarily on detection tasks. While initial results on cross-domain object detection benchmarks are promising, we have not yet validated the framework on dense prediction tasks such as segmentation or regression-based tasks like depth estimation. Extending to these settings could further validate generality.

Failure cases. Typical failure modes include (i) missing extremely small cracks occupying less than one patch, and (ii) false positives triggered by strong shadows or painted markings. These highlight the need for higher-resolution feature extraction or domain-specific regularization strategies.

J.2. Societal Impact

Positive impact. Robust automated road inspection systems can directly enhance public safety by enabling early detection of hazardous defects, preventing accidents, and guiding timely maintenance. Economically, municipalities can benefit from more efficient allocation of repair budgets, reducing long-term infrastructure costs. Environmentally, extending pavement lifespans through proactive maintenance reduces the need for energy-intensive repaving.

Ethical considerations. Automation raises potential workforce displacement for human inspectors. It is crucial to develop retraining programs to transition affected workers into complementary roles such as system oversight, quality control, or data analysis. Furthermore, large-scale data collection (e.g., street-level imagery) raises privacy concerns. Deployments must ensure anonymization (e.g., face and license plate blurring) and compliance with data protection regulations. Finally, algorithmic bias remains a concern: if training data over-represents certain geographies or road types, models may underperform in under-represented regions, leading to inequities in infrastructure maintenance.

J.3. Future Work

Advanced prompt generation. Moving beyond static K-means clustering, future work could explore end-to-end prompt generation mechanisms, such as deep clustering integrated with contrastive learning, or graph-based prototype discovery. This could yield prompts that are both semantically rich and directly optimized for downstream tasks.

Source-free and open-set adaptation. A promising extension is *source-free DA*, where only the pretrained source model is available at adaptation time. Another direction is *open-set DA*, where new defect categories appear in target domains. Integrating uncertainty estimation, open-set recognition, and incremental prompt learning could make the framework more adaptive in realistic deployments.

Integration with vision-language models. Recent progress in large vision-language models (VLMs) suggests the possibility of generating prompts guided by textual descriptions (e.g., "longitudinal cracks" or "circular potholes"). Such multimodal prompting could enable more controllable and interpretable adaptation, bridging computer vision with domain expert knowledge.

Applications beyond defect detection. The principles underlying PROBE —learning target-aware prompts and aligning them across domains—are not task-specific. Potential applications include bridge crack inspection, corrosion detection in industrial pipelines, visual quality control in manufacturing, and medical imaging (e.g., cross-hospital domain shifts in CT or MRI scans).

Human-in-the-loop adaptation. Given PROBE 's strong zero-shot performance, it is well-suited as a foundation for active learning. A future system could automatically highlight uncertain detections and request annotations from human experts. This selective labeling strategy would further reduce annotation cost and improve adaptability to new conditions.

J.4. Summary

In summary, PROBE advances the frontier of self-supervised prompting for domain adaptation but is not with-

out limitations. By acknowledging these challenges and outlining future research avenues, we aim to provide a roadmap for building truly robust, efficient, and socially responsible cross-domain vision systems.