VARMATCH: VARIATIONAL CONSISTENCY AND TEST-TIME ADAPTATION FOR SEMI-SUPERVISED DOMAIN GENERALIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Semi-supervised domain generalization (SSDG) aims to generalize to unseen domains using only a few labeled examples per class and additional unlabeled data from multiple source domains. In this setting, limited supervision makes it difficult to learn reliable class structure, while domain shift can cause the learned structure to become misaligned with target data. We introduce VarMatch, a variational latent-space framework that combines stochastic consistency training with source-aware and frozen test-time adaptation. During training, pseudo-labels obtained from weakly augmented examples supervise two complementary views: an image-space augmentation and a variance-scaled stochastic perturbation of the learned latent representation. This encourages predictions to remain consistent under both input-level and representation-level variations. At test time, VarMatch keeps the encoder and classifier frozen, aligns target representations with source class structure, and estimates target-aware class prototypes from confident target samples while excluding samples that provide little evidence of domain shift. Adaptation operates without network-parameter updates and uses deterministic latent representations. Across PACS, OfficeHome, Digits-DG, and miniDomainNet with 5 and 10 labels per class, VarMatch improves over the strongest semi-supervised domain-generalization baseline by 2.37-6.22 percentage points. Its frozen, source-aware adaptation outperforms evaluated parameter-free alternatives and is competitive with methods that update the encoder.

1 INTRODUCTION

Domain generalization (DG) considers the problem of training a model on several source domains and generalizing to a target domain that is never observed during training. The semi-supervised setting (SSDG) makes this problem particularly challenging: only a handful of samples per class are labeled in each source domain, while the remaining source data is unlabeled (Zhou et al., 2023). The goal is therefore not only to learn from multiple domains, but to learn reliable class-relevant structure from very limited supervision while avoiding domain-specific information that may not transfer to an unseen target domain.

Two difficulties arise in this regime. First, with few labels per class, the supervised examples provide only a weak estimate of the class structure used by the classifier. The much larger amount of unlabeled source data is therefore essential, but exploiting it without allowing noisy predictions to corrupt training remains difficult. Semi-supervised learning addresses this through pseudo-labeling and consistency regularization: predictions from weak views are used as supervision for strongly augmented views when their confidence is sufficiently high (Sohn et al., 2020; Zhang et al., 2021; Chen et al., 2023; Wang et al., 2023). Some SSDG methods further extend this process through style transfer (Zhou et al., 2023), refining the use of unlabeled examples through feature-space semantic alignment (Galappaththige et al., 2024), or by explicitly utilizing unlabeled examples that fall below the confidence threshold (Lee et al., 2025). However, these approaches construct consistency views through predefined input-space or style transformations, rather than from a distribution over representations that the model can learn for each sample.

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

Second, a classifier trained on source domains may induce misplaced class boundaries on an unseen target domain. Even when class identities are shared, target samples can exhibit class-dependent shifts in location and scale, causing their latent representations to become misaligned with the source class centroids and classifier weights. This is particularly problematic when the underlying class structure was estimated from only a few labeled examples.

We present *VarMatch*, a variational latent-space framework that addresses these challenges through a shared posterior representation at training and test time. During training, a weak-view pseudo-label supervises two complementary views: an image-space augmentation and a variance-scaled stochastic perturbation of the sample's learned latent mean. The latter uses the uncertainty encoded by the posterior variance to control the magnitude of latent variation, providing an additional consistency signal directly in the representation space used for prediction. At test time, this learned latent representation provides the basis for source-aware adaptation while the network remains frozen. VarMatch first applies a class-conditional latent transformation to align target latent means with the source class geometry, and then replaces source prototypes in backbone feature space with centroids of confident target samples that are sufficiently distinct from source-domain examples. Training uses only labeled and unlabeled source data, while test-time adaptation uses source information together with unlabeled target data.

Our contributions are as follows.

- We introduce an uncertainty-conditioned latent consistency mechanism for SSDG: a learned per-sample variational posterior modulates a stochastic latent view using its variance and prediction confidence. This adapts the consistency transformation itself and, on unlabeled source data, improves pseudo-label accuracy and class separation.
- We develop a frozen, source-aware test-time correction that combines class-conditional latent alignment with proximity-filtered prototype replacement, using confident target samples only when they provide geometric evidence beyond the source representation.
- VarMatch improves over prior SSL and SSDG baselines (Tables 1 and 2) across PACS, OfficeHome, Digits-DG, and miniDomainNet at both 5 and 10 labels per class. Furthermore, when compared with established test-time adaptation methods, our frozen checkpoints combined with the source-aware test-time procedure outperform these methods, matching the performance of methods that require updating the encoder on target data (Table 4).

2 Related Work

2.1 Background: The Variational Information Bottleneck

The information bottleneck (IB) (Tishby et al., 1999) seeks a representation Z that is maximally informative about the label Y while minimizing information about the input X, formulated as $\max_{p(z|x)} I(Z;Y) - \beta I(Z;X)$. Since the mutual information terms are generally intractable for deep networks, Alemi et al. (2017) introduce a variational posterior $q(\boldsymbol{z} \mid x) = \mathcal{N}\big(\boldsymbol{\mu}(x), \mathrm{diag}(\boldsymbol{\sigma}^2(x))\big)$ and prior $r(\boldsymbol{z}) = \mathcal{N}(0, I)$, yielding the objective $\mathcal{L}_{\mathrm{VIB}} = \mathbb{E}_{(x,y)}\Big[\mathbb{E}_{\boldsymbol{z}\sim q(\boldsymbol{z}|x)}\big[-\log p(y \mid \boldsymbol{z})\big] + \beta\, D_{\mathrm{KL}}\big(q(\boldsymbol{z} \mid x) \,\|\, r(\boldsymbol{z})\big)\Big]$, optimized using the reparameterization $\boldsymbol{z} = \boldsymbol{\mu}(x) + \boldsymbol{\sigma}(x) \odot \boldsymbol{\epsilon}$ with $\boldsymbol{\epsilon} \sim \mathcal{N}(0, I)$ (Kingma & Welling, 2014). In standard VIB, stochastic samples from $q(\boldsymbol{z} \mid x)$ are used during training, while the posterior mean $\boldsymbol{\mu}(x)$ is commonly used for deterministic inference. VarMatch follows that inference convention, and additionally uses the posterior scale during training as a second consistency view (Sec. 3).

2.2 Semi-Supervised Learning and Consistency Regularization

Modern semi-supervised learning (SSL) commonly combines pseudo-labeling with consistency regularization between weakly and strongly augmented views. FixMatch (Sohn et al., 2020) generates a pseudo-label from a weakly augmented sample, retains it only above a confidence threshold, and uses it to supervise a strongly augmented view. Other methods improve the use of unlabeled data through augmentation-based consistency training and distribution regularization (Berthelot et al., 2019; 2020; Xie et al., 2020). To overcome the limitations of a static threshold, subsequent methods dynamically refine sample selection: FlexMatch (Zhang et al., 2021) introduces class-specific

curriculum thresholds, FreeMatch (Wang et al., 2023) employs self-adaptive thresholding via prediction moving averages, and SoftMatch (Chen et al., 2023) replaces hard cutoffs with continuous, confidence-based Gaussian soft weights. These methods primarily construct consistency views in input space. VarMatch retains this input-space consistency mechanism and adds a second view generated in latent space from the learned posterior.

2.3 Domain Generalization

Domain generalization (DG) aims to learn from one or more source domains and generalize to unseen target domains. Beyond empirical risk minimization (ERM), which minimizes the average training loss across source domains (Vapnik, 1998), existing approaches address domain shift through invariant representations (e.g., Arjovsky et al., 2019), information-bottleneck approaches such as MetaVIB (Du et al., 2020) and IIB (Li et al., 2022a), feature-level regularization such as RSC (Huang et al., 2020) that iteratively discards dominant training features, and flatness-aware optimization (e.g., Cha et al., 2021). These methods generally assume fully labeled source domains, whereas the SSDG setting considered by VarMatch has only a few labeled examples per class, making reliable use of unlabeled data an integral part of generalization across domains.

2.4 Semi-supervised Domain Generalization

Semi-supervised domain generalization (SSDG) considers multiple source domains with limited labeled data and additional unlabeled samples. Existing methods tackle this challenge through diverse strategies. At the data level, Zhou et al. (2023) introduced StyleMatch, extending FixMatch with a stochastic classifier and a style-based consistency view generated by a pretrained style-transfer network. At the representation level, MultiMatch (Qi et al., 2024) formulates SSDG as a multi-task learning problem across source domains, while FBC-SA (Galappaththige et al., 2024) introduces feature conformity and semantic alignment. Other recent approaches focus specifically on overcoming unreliable pseudo-labels: CAT (Zoha et al., 2025) addresses this through class-aware adaptive thresholds and neighborhood-based refinement, while UPCSC (Lee et al., 2025) leverages low-confidence unlabeled samples via proxy-based contrastive and surrogate-class learning. VarMatch takes a complementary approach. Rather than relying on external style models or cross-domain alignment, it exploits the variational posterior. This allows VarMatch to construct an intrinsic latent-space consistency view during training, while uniquely retaining its learned geometry for test-time adaptation.

2.5 Latent-Space Augmentation

Beyond input-space transformations, several works construct augmentations or perturbations directly in feature space. ISDA (Wang et al., 2019) samples semantic directions from class-conditional covariance estimates, while FeatMatch (Kuo et al., 2020) uses prototypical representations for feature refinement and augmentation in semi-supervised learning. Additionally, LVAT (Osada et al., 2020) applies adversarial perturbations in the latent space of a generative model and decodes them into images. For domain generalization, MixStyle (Zhou et al., 2021) mixes feature statistics across instances, while DSU (Li et al., 2022b) models feature statistics as uncertain distributions. In contrast, VarMatch constructs its second view by modulating the latent mean with the per-sample posterior uncertainty $q(\boldsymbol{z} \mid x)$, without auxiliary feature statistics or a separate generator.

2.6 Test-Time Adaptation

Test-time adaptation (TTA) leverages unlabeled target samples at inference to mitigate distribution shift. Gradient-based methods such as TENT (Wang et al., 2021) update selected model parameters through entropy minimization (Niu et al., 2022; 2023). Source-free methods such as SHOT (Liang et al., 2020) optimize the feature extractor while keeping the source classifier fixed. Parameter-free approaches instead avoid parameter updates by directly adapting predictions or classifier outputs (e.g., Iwasawa & Matsuo, 2021; Boudiaf et al., 2022). Source-aware methods retain source information, such as source features or prototypes, during adaptation (Zhang et al., 2023; Karimi & Dibeklioglu, 2025). VarMatch is a frozen, source-aware method: it

keeps the network parameters fixed and adapts using latent means and backbone features, together with source information and unlabeled target data.

3 METHODOLOGY

VarMatch leverages the latent representation space for both training-time consistency regularization and test-time domain adaptation. The architecture consists of: (1) a convolutional backbone $f(\cdot)$ that maps input images to feature representations $\boldsymbol{h}$, (2) a variational layer that maps $\boldsymbol{h}$ to a diagonal Gaussian posterior $\mathcal{N}(\boldsymbol{\mu}, \mathrm{diag}(\boldsymbol{\sigma}^2))$, regularized toward a standard normal prior through the information bottleneck, and (3) a stochastic cosine classifier $C(\cdot)$ operating on normalized latent codes.

Building on FixMatch (Sohn et al., 2020), VarMatch trains on labeled source data with a supervised cross-entropy $\mathcal{L}_{\mathrm{labeled}}$ and enforces consistency on unlabeled data across two complementary views (Fig. 1): a strongly augmented image view, supervised by $\mathcal{L}_{\mathrm{strong}}$, and a *variance-scaled resampling* (VSR) view drawn from the learned posterior, supervised by $\mathcal{L}_{\mathrm{vsr}}$. At test time all network parameters remain frozen: a lightweight Latent Transformation (LT) aligns target latent means with the source class structure, and confident target samples, after a source-proximity filter, form a support set $\mathcal{S}$ used for prototype interpolation in the backbone feature space. The remainder of this section details the training objective (Sec. 3.1), LT (Sec. 3.2), and proximity-filtered prototype interpolation (Sec. 3.3).

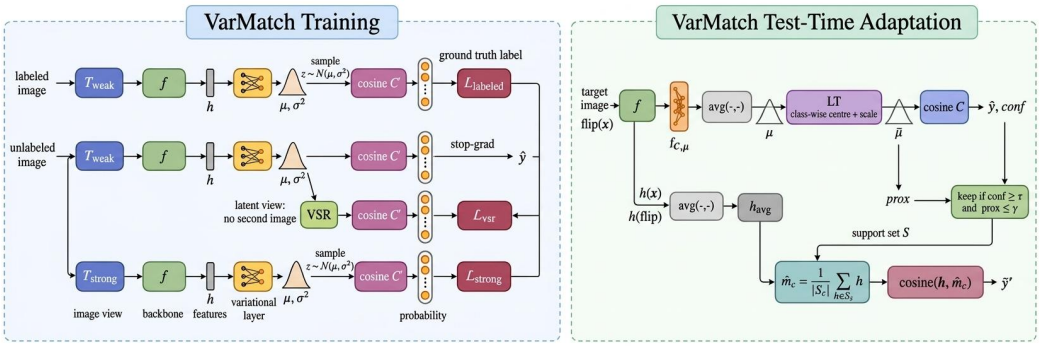


Figure 1: Overview of VarMatch during training and test-time evaluation.

3.1 SOURCE TRAINING FRAMEWORK

Setup and Notation. We follow the standard SSDG protocol: training uses K source domains with a small labeled set $\mathcal{X}$ and an unlabeled set $\mathcal{U}$, and evaluation is on an unseen target domain. A backbone f maps an image x to features $\boldsymbol{h} = f(x)$, from which two heads f_μ, f_σ parameterize a diagonal Gaussian posterior over a latent code,

$$\boldsymbol{\mu} = f_\mu(\boldsymbol{h}), \qquad \log \boldsymbol{\sigma}^2 = f_\sigma(\boldsymbol{h}), \qquad \boldsymbol{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(0, I), \tag{1}$$

and we refer to f together with f_μ, f_σ as the encoder. The latent dimension matches that of $\boldsymbol{h}$, so the bottleneck is imposed by the KL term rather than by a compressed dimension. The diagonal covariance keeps the KL to $\mathcal{N}(0, I)$ in closed form, $\mathcal{L}_{\mathrm{KL}} = \frac{1}{2}\sum_j(\mu_j^2 + \sigma_j^2 - \log \sigma_j^2 - 1)$, and provides a per-dimension uncertainty estimate $\boldsymbol{\sigma}(x)$. Predictions use a temperature-scaled cosine classifier $C(\boldsymbol{v})_c = \cos(\boldsymbol{v}, \boldsymbol{w}_c)/T_{\mathrm{cls}}$, with $\boldsymbol{w}_c$ the learnable weight for class c. We adopt the stochastic classifier of Zhou et al. (2023), sampling weights around $\boldsymbol{w}_c$ in $\mathcal{L}_{\mathrm{labeled}}$ and $\mathcal{L}_{\mathrm{strong}}$; pseudo-label generation, $\mathcal{L}_{\mathrm{vsr}}$ and inference use the mean $\boldsymbol{w}_c$. Classifier sampling and weight updates are detailed in Appendix A.

Pseudo-Label Generation and Filtering. Given an unlabeled image u, we forward its weakly augmented version to obtain the latent mean $\boldsymbol{\mu}$ and score it against the class weights $\boldsymbol{w}_c$ under

cosine similarity $\mathrm{sim}(\cdot,\cdot)$. The pseudo-label distribution mixes two temperatures of these scores:

$$q(u) = (1-\eta)\,\mathrm{softmax}\big(\mathrm{sim}(\boldsymbol{\mu}, \boldsymbol{w}_c)/T_{\mathrm{cls}}\big) + \eta\,\mathrm{softmax}\big(\mathrm{sim}(\boldsymbol{\mu}, \boldsymbol{w}_c)/T_{\mathrm{proto}}\big). \tag{2}$$

Since both terms are strictly increasing transformations of the same similarity scores, they have the same argmax. Therefore, any positively weighted mixture of the two preserves the predicted class while mildly smoothing the confidence scores. The pseudo-label $\hat{q}(u) = \arg\max_c q_c(u)$ is computed without gradient.

A pseudo-label is retained if $\max_c q_c(u)$ exceeds a confidence threshold. We use either FixMatch's fixed $\tau = 0.95$ (Sohn et al., 2020) or the class-adaptive threshold of FreeMatch (Wang et al., 2023), $\tau_c = \tau_g E(c)/\max_{c'} E(c')$. Here τ_g is an exponential moving average (EMA) of the mean confidence over all samples, and $E(c)$ is the EMA restricted to samples assigned to class c. Unlike FreeMatch, both averages include labeled as well as unlabeled samples, and $E(c)$ tracks the mean confidence of samples assigned to c rather than the mean probability of class c. Which rule is used depends on the number of classes (Sec. 4). Let m denote the resulting binary mask, shared by the two perturbed views.

One Pseudo-Label, Two Perturbed Views. The pseudo-label supervises two views constructed in different representation spaces. The first is an image-space view: u is strongly augmented (Xie et al., 2020), re-encoded to $(\boldsymbol{\mu}^{\mathrm{s}}, \boldsymbol{\sigma}^{\mathrm{s}})$, and classified from $\boldsymbol{z}^{\mathrm{s}} \sim \mathcal{N}\big(\boldsymbol{\mu}^{\mathrm{s}}, \mathrm{diag}((\boldsymbol{\sigma}^{\mathrm{s}})^2)\big)$. The second re-encodes the same weakly augmented image and scales the posterior sampling radius via variance-scaled resampling (VSR):

$$\boldsymbol{z}^{\mathrm{vsr}} = \boldsymbol{\mu} + s(u)\,\boldsymbol{\sigma} \odot \boldsymbol{\epsilon}, \qquad s(u) = s_0\big(\max_c q_c(u)\big)^{\omega}, \tag{3}$$

where s_0 is the base perturbation radius and ω controls the strength of confidence-based scaling. Writing $p(c \mid \boldsymbol{z}) = \mathrm{softmax}(C(\boldsymbol{z}))_c$, both views use the shared pseudo-label with the confidence gate m:

$$\mathcal{L}_{\mathrm{strong}} = -m \log p\big(\hat{q}(u) \mid \boldsymbol{z}^{\mathrm{s}}\big), \qquad \mathcal{L}_{\mathrm{vsr}} = -m \cdot \max_c q_c(u) \log p\big(\hat{q}(u) \mid \boldsymbol{z}^{\mathrm{vsr}}\big). \tag{4}$$

Unlike the fixed image-space perturbation, VSR perturbs the latent code along $\boldsymbol{\sigma}(x)$, the per-dimension uncertainty learned by the variational encoder. Since $\boldsymbol{\mu}$ remains the anchor, VSR does not bias the representation toward a class center, while $s(u)$ reduces the perturbation for lower-confidence samples. The latent view is thus softly masked, confidence weighting its loss among retained samples, while the image view uses the binary mask alone.

Auxiliary Losses. To regularize the latent geometry, we apply supervised contrastive learning (Khosla et al., 2020) to labeled and confident unlabeled samples (using weak views and pseudo-labels $\hat{q}(u)$). Let $\mathcal{I}$ denote these samples, $\mathcal{P}(i)$ the positives for anchor i, and $\mathcal{A}(i)$ all other batch samples. The loss is

$$\mathcal{L}_{\mathrm{supcon}} = -\frac{1}{|I^+|} \sum_{i \in I^+} \frac{1}{|P(i)|} \sum_{p \in P(i)} \log \frac{\exp\big(\mathrm{sim}(\mu_i, \mu_p)/T_{\mathrm{con}}\big)}{\sum_{a \in A(i)} \exp\big(\mathrm{sim}(\mu_i, \mu_a)/T_{\mathrm{con}}\big)}, \tag{5}$$

where $\mathcal{I}^+ = \{i \in \mathcal{I} : |\mathcal{P}(i)| > 0\}$. For unconfident unlabeled samples, assuming N_u such samples in the batch, we compute an NT-Xent loss (Chen et al., 2020) across their $2N_u$ weak and strongly augmented image views, where $p(i)$ denotes the paired view:

$$\mathcal{L}_{\mathrm{ntxent}} = -\frac{1}{2N_u} \sum_{i=1}^{2N_u} \log \frac{\exp\big(\mathrm{sim}(\mu_i, \mu_{p(i)})/T_{\mathrm{con}}\big)}{\sum_{k \neq i} \exp\big(\mathrm{sim}(\mu_i, \mu_k)/T_{\mathrm{con}}\big)}. \tag{6}$$

In addition, a center-pull loss encourages the latent means of labeled samples to remain close to their cosine-classifier weights, $\mathcal{L}_{\mathrm{pull}} = \sum_{k=1}^{K} \frac{1}{|\mathcal{X}_k| d} \sum_{i \in \mathcal{X}_k} \|\boldsymbol{\mu}_i - \mathrm{sg}(\boldsymbol{w}_{y_i})\|_2^2$, where $\mathcal{X}_k \subset \mathcal{X}$ denotes the labeled samples of source domain k in the batch, d is the latent dimension, and $\mathrm{sg}(\cdot)$ is a stop-gradient; $\boldsymbol{w}_{y_i}$ is the unnormalized weight, since normalization happens only inside C. The total auxiliary objective is combined as $\mathcal{L}_{\mathrm{aux}} = \lambda_{\mathrm{con}}(\mathcal{L}_{\mathrm{supcon}} + \mathcal{L}_{\mathrm{ntxent}}) + \lambda_{\mathrm{pull}}\mathcal{L}_{\mathrm{pull}}$.

Objective. VarMatch optimizes the combined objective

$$\mathcal{L} = \mathcal{L}_{\mathrm{labeled}} + \beta \mathcal{L}_{\mathrm{KL}} + \lambda_u\,(\mathcal{L}_{\mathrm{strong}} + \mathcal{L}_{\mathrm{vsr}}) + \mathcal{L}_{\mathrm{aux}}, \tag{7}$$

where $\mathcal{L}_{\mathrm{KL}}$ is evaluated on both labeled and unlabeled data. We use β to control the strength of the variational regularization and λ_u to weight the unlabeled consistency terms.

3.2 TEST-TIME LATENT TRANSFORMATION

At test time, source and target samples may exhibit different class-wise geometry in the latent space. Latent Transformation estimates this shift from the unlabeled target set and maps target representations to the corresponding source geometry. LT consists of a fitting stage followed by inference.

Fitting. A forward pass over the source data provides latent means $\boldsymbol{\mu}^{\mathrm{src}}$. For each class c, we compute a source centroid $\boldsymbol{m}_c^{\mathrm{src}}$ from labeled samples and high-confidence unlabeled source samples (confidence ≥ 0.95), while storing labeled representations and at most N high-confidence unlabeled representations per class in a memory bank $\mathcal{B}$ for the proximity computation in Sec. 3.3. We also compute the source global mean $\bar{\boldsymbol{\mu}}^{\mathrm{src}}$ over all labeled and unlabeled source samples.

We then perform a single forward pass over the target set and store its latent means $\boldsymbol{\mu}_j^{\mathrm{tgt}}$, $j = 1, \ldots, M$. LT operates only on these stored representations during fitting, so images are not re-encoded at each iteration. We initialize the target global mean as $\bar{\boldsymbol{\mu}}^{\mathrm{tgt}} = \mathrm{mean}_{j=1}^{M} \boldsymbol{\mu}_j^{\mathrm{tgt}}$. At each iteration, the stored representations are shifted into the source frame as $\boldsymbol{\mu}'_j = \boldsymbol{\mu}_j^{\mathrm{tgt}} - \bar{\boldsymbol{\mu}}^{\mathrm{tgt}} + \bar{\boldsymbol{\mu}}^{\mathrm{src}}$. Each shifted representation $\boldsymbol{\mu}'_j$ is assigned a class $\hat{y}_j = \arg\max_c C(\boldsymbol{\mu}'_j)_c$ and a confidence $\kappa_j = \max_c \mathrm{softmax}(C(\boldsymbol{\mu}'_j))_c$; the target centroid $\boldsymbol{m}_c^{\mathrm{tgt}}$ is then the κ-weighted mean of the original $\boldsymbol{\mu}_j^{\mathrm{tgt}}$ among the most confident samples assigned to class c. The target global mean is updated toward the mean of the resulting class centroids, and the procedure is repeated until convergence. Finally, each class receives a scale ratio s_c given by the ratio of its median within-class source and target distances.

Inference. For a target image, one or two forward passes, including a horizontal flip when used, produce latent means that are averaged to obtain $\boldsymbol{\mu}$. We first obtain a class assignment $\hat{y} = \arg\max_c C(\boldsymbol{\mu})_c$, which determines the target and source class statistics used by LT. The representation is then transformed as

$$\tilde{\boldsymbol{\mu}} = s_{\hat{y}} \left(\boldsymbol{\mu} - \boldsymbol{m}_{\hat{y}}^{\mathrm{tgt}} \right) + \boldsymbol{m}_{\hat{y}}^{\mathrm{src}}. \tag{8}$$

The class is re-evaluated after transformation, and Eq. 8 is reapplied to the original $\boldsymbol{\mu}$ using the updated assignment. Reapplying the map reduces sensitivity to an incorrect initial class prediction: the first transformation refines the class assignment, while the second uses the revised class statistics but operates on the original representation, thereby avoiding the compounding of transformations. The resulting assignment and confidence supply the pseudo-labels used by prototype interpolation in Sec. 3.3. Implementation details and edge cases are given in Appendix A.1.

3.3 TEST-TIME PROTOTYPE INTERPOLATION (TT-PI)

LT aligns target latent representations to the source geometry, but the cosine classifier still uses source-trained class weights $\boldsymbol{w}_c$. TT-PI addresses the remaining classifier–target mismatch by estimating class prototypes directly from confident target features, without updating the network parameters. Thus, LT provides the initial pseudo-labels and latent alignment, while TT-PI performs a test-time replacement of the source prototypes for the final prediction.

For each target sample x_j, let $\hat{y}_j$ and κ_j denote the class prediction and confidence from $C(\tilde{\boldsymbol{\mu}}_j)$ produced by LT. We form the support set

$$\mathcal{S} = \{ j : \kappa_j \geq \tau_{\mathrm{tt}} \ \wedge \ \mathrm{prox}(j) \leq \gamma \}, \tag{9}$$

where $\tau_{\mathrm{tt}} = 0.90$ and $\mathrm{prox}(j)$ measures similarity to the source memory bank $\mathcal{B}$ of Sec. 3.2. The threshold γ is the q-quantile of the target proximity distribution with $q = 0.8$, computed from unlabeled target proximities alone; implementation details are given in Appendix A.2.

For each class c, selected target samples are averaged in the pre-bottleneck backbone space as $\hat{\boldsymbol{m}}_c^{\mathrm{tgt}} = \frac{1}{|\mathcal{S}_c|} \sum_{j \in \mathcal{S}_c} \boldsymbol{h}_j$, where $\mathcal{S}_c = \{ j \in \mathcal{S} : \hat{y}_j = c \}$. As with $\boldsymbol{\mu}$, $\boldsymbol{h}_j$ is averaged over the image and its flip when used. We use h rather than $\tilde{\mu}$ for prototype construction, since h is not constrained by the latent KL regularization, while $\tilde{\mu}$ is retained for the original classifier and support selection.

Target prototypes are interpolated with the source prototypes,

$$\hat{\boldsymbol{m}}_c = (1-\alpha)\,\hat{\boldsymbol{m}}_c^{\mathrm{src}} + \alpha\,\hat{\boldsymbol{m}}_c^{\mathrm{tgt}} \quad \text{if } |\mathcal{S}_c| > 0, \qquad \hat{\boldsymbol{m}}_c = \hat{\boldsymbol{m}}_c^{\mathrm{src}} \quad \text{otherwise,} \tag{10}$$

where $\hat{\boldsymbol{m}}_c^{\mathrm{src}}$ is the mean backbone feature of labeled source examples of class c. We use $\alpha = 1$, so target prototypes replace source prototypes wherever target support exists. The final prediction is $y_j^* = \arg\max_c \cos(\boldsymbol{h}_j, \hat{\boldsymbol{m}}_c)$.

4 Experiments

Datasets and protocol. We evaluate on four SSDG benchmarks: PACS (7 classes), OfficeHome (OH, 65 classes), Digits-DG (10 classes) and miniDomainNet (DN, 126 classes), each comprising four domains. Each domain is held out as the unseen target in turn while the remaining three act as sources, with 5 or 10 labeled images per class per source domain and all remaining source images treated as unlabeled. For PACS and OH we adopt the splits of Zhou et al. (2023), and for Digits-DG and DN those of Lee et al. (2025). [1] Results are top-1 accuracy on the target domain, averaged over the four leave-one-domain-out splits and three random seeds.

Implementation details. We use an ImageNet-pretrained ResNet-18 with the variational encoder and cosine classifier of Sec. 3, optimized with SGD (momentum 0.9, weight decay 5×10^{-3}) and a cosine schedule. Batches contain equal numbers of labeled and unlabeled images. We use the class-adaptive threshold of Sec. 3.1 on PACS and Digits-DG and a fixed $\tau = 0.95$ on OfficeHome and miniDomainNet. Horizontal flipping is omitted on Digits-DG at both training and test time. We set $\lambda_{\mathrm{con}} = 0.05$, $\lambda_{\mathrm{pull}} = 0.02$, and $\lambda_u = 1$. Training uses fixed per-benchmark schedules and a uniform weight average of the final epoch checkpoints (Appendix B).

Baselines. We compare against ERM (Vapnik, 1998) as a representative DG method; the SSL methods FreeMatch (Wang et al., 2023) and FixMatch (Sohn et al., 2020); the SSDG method StyleMatch (Zhou et al., 2023); and the plug-and-play modules FBC-SA (Galappaththige et al., 2024) and UPCSC (Lee et al., 2025) attached to each SSL baseline. All baseline figures are quoted from Lee et al. (2025), who evaluate every method under a single protocol. We adopt the same splits, labeling budgets, ImageNet-pretrained ResNet-18 backbone, and the same Dassl evaluation code (Zhou et al., 2023). Those figures are averaged over five seeds, ours over three.

Results. Tables 1 and 2 report target-domain accuracy at the two label budgets. Relative to the strongest compared configuration in each column, VarMatch improves by 4.82, 5.55, 2.37, and 3.73 points at 10 labels per class, and by 5.66, 5.51, 6.22, and 2.96 points at 5. The gains hold at both budgets and on every benchmark, from the 7-class PACS to the 126-class miniDomainNet. The largest single gain, 6.22 points, occurs on Digits-DG at 5 labels, where the compared methods degrade most sharply relative to their 10-label results.

4.1 Replacing the Test-Time Stage

We replace the VarMatch test-time stage with established TTA methods on the same frozen checkpoints, using the uniform weight average of the last 5 epoch checkpoints to isolate the adaptation mechanism from training. All methods are evaluated at 10 labels per class over the four leave-one-domain-out splits and three seeds. The raw model denotes the VarMatch-trained model evaluated without any test-time adaptation. Each baseline uses the authors' official implementation with its published settings; SHOT is run for 15 target epochs. LAME is reported with both its default RBF and kNN affinities. AdaNPC uses the official kNN update. Unlabeled source images have no ground-truth labels in this protocol, so its memory is the labeled set plus the top 100 confident unlabeled samples per class, the same source set used by our latent transformation. VarMatch additionally averages predictions over a horizontal flip, which the baseline methods do not use. [2]

[1] On DN, a few classes have fewer images than the budget in some source domains, so the labeled set is slightly smaller than $126 \times 3 \times 10$ (or $126 \times 3 \times 5$).

[2] T3A, LAME, TENT, and AdaNPC adapt incrementally over target batches and use a shuffled test sampler; SHOT and VarMatch operate over the full target set.

Table 1: Target-domain accuracy (%) at 10 labels per class. Baselines are from Lee et al. (2025); VarMatch reports mean ± std across training seeds. Best per column in bold, second best underlined.

Method	PACS	OfficeHome	Digits-DG	miniDomainNet
ERM	60.2 ± 2.0	54.2 ± 0.5	60.8 ± 3.1	48.8 ± 0.2
FreeMatch	73.5 ± 1.1	57.7 ± 0.4	74.2 ± 2.1	54.8 ± 0.2
FixMatch	76.8 ± 1.1	57.7 ± 0.4	75.1 ± 1.1	54.5 ± 0.3
StyleMatch	79.9 ± 1.0	59.7 ± 0.3	78.4 ± 0.4	55.0 ± 0.2
FixMatch + FBC-SA	77.7 ± 1.6	58.7 ± 0.4	80.5 ± 1.3	55.6 ± 0.3
StyleMatch + FBC-SA	79.3 ± 3.0	<u>60.0 ± 0.3</u>	80.5 ± 1.4	55.0 ± 0.3
FreeMatch + UPCSC	77.8 ± 1.4	59.1 ± 0.5	80.4 ± 0.7	<u>56.5 ± 0.3</u>
FixMatch + UPCSC	79.6 ± 0.7	58.6 ± 0.2	80.7 ± 1.1	56.0 ± 0.3
StyleMatch + UPCSC	<u>81.5 ± 1.2</u>	59.9 ± 0.1	<u>82.2 ± 0.6</u>	55.6 ± 0.3
VarMatch (ours)	**86.32 ± 0.20**	**65.55 ± 0.30**	**84.57 ± 0.49**	**60.23 ± 0.39**

Table 2: Target-domain accuracy (%) at 5 labels per class; conventions as in Table 1.

Method	PACS	OfficeHome	Digits-DG	miniDomainNet
ERM	55.2 ± 2.5	52.2 ± 0.6	42.6 ± 2.7	44.4 ± 0.3
FreeMatch	71.6 ± 1.8	55.9 ± 0.5	63.3 ± 2.8	52.0 ± 0.7
FixMatch	73.6 ± 2.9	55.0 ± 0.6	64.7 ± 3.8	51.6 ± 0.3
StyleMatch	78.9 ± 0.8	56.5 ± 0.5	71.9 ± 2.9	51.0 ± 0.4
FixMatch + FBC-SA	74.2 ± 2.9	55.6 ± 0.4	75.5 ± 0.9	50.1 ± 0.2
StyleMatch + FBC-SA	76.8 ± 2.6	55.8 ± 0.3	75.8 ± 5.7	50.1 ± 0.2
FreeMatch + UPCSC	73.5 ± 2.1	<u>56.8 ± 0.8</u>	76.4 ± 0.6	<u>53.7 ± 0.4</u>
FixMatch + UPCSC	78.9 ± 0.9	56.1 ± 0.6	75.2 ± 2.6	52.7 ± 0.4
StyleMatch + UPCSC	<u>79.8 ± 3.2</u>	<u>56.8 ± 0.8</u>	<u>76.7 ± 2.1</u>	51.2 ± 0.2
VarMatch (ours)	**85.46 ± 0.77**	**62.31 ± 0.40**	**82.92 ± 1.93**	**56.66 ± 0.27**

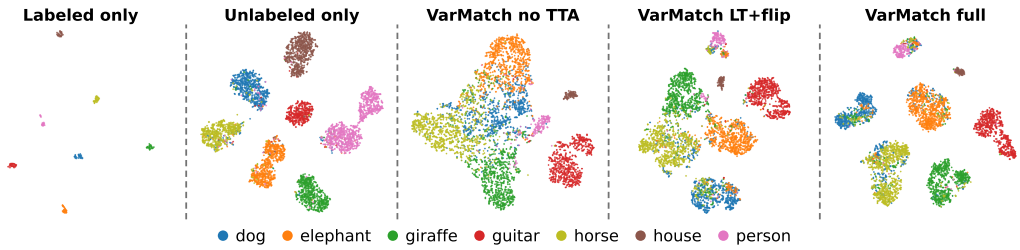


Figure 2: t-SNE of the representations on PACS with Sketch as the unseen target, at 10 labels per class, colored by ground-truth class. The first four panels show latent means μ: the labeled source set, the unlabeled source set, the target domain under the trained model, and the target domain after latent transformation and flip-averaging. The last panel shows the target backbone features h on which prototype replacement operates.

Table 4 reports the results. The VarMatch test-time stage improves the raw model by 4.30 points on PACS and 3.72 on OfficeHome, achieving the highest accuracy on PACS; Figure 2 shows the source and target representations before and after the test-time stage. On OfficeHome, VarMatch and SHOT obtain similar mean accuracy, although the comparison is not strictly equivalent: SHOT updates the encoder, whereas VarMatch keeps it frozen but retains source features. Both operate over the complete target set. Among the frozen baselines, none improves over the raw model by more than 1.75 points; T3A falls below the raw model on OfficeHome, while LAME's default RBF affinity performs substantially worse in this setting; its kNN variant, which keeps only nearest neighbors on the same features, remains close to the raw model.

Table 3: PACS ablations at 10 labels per class, three-seed mean ± sample std. All rows are retrained. Full denotes the complete training configuration and matches Table 1.

Method	PACS
VarMatch Full	86.32 ± 0.20
− image augmentation	82.29 ± 1.25
− variational view	85.90 ± 0.56
− adaptive threshold	86.03 ± 0.59
− pseudo-label smoothing ($\eta = 0$)	86.03 ± 0.18
− center-pull	86.63 ± 0.26
− SupCon	86.67 ± 0.41
− SupCon − center-pull	86.56 ± 0.40
− all auxiliaries	85.98 ± 0.34
− adaptive threshold − SupCon	85.81 ± 0.12

Table 4: TTA methods on the same VarMatch checkpoints, 10 labels per class, three-seed mean ± sample std. Best in bold, second best underlined.

Method	PACS	OfficeHome
Raw model	82.02 ± 0.53	61.83 ± 0.39
Frozen parameters		
T3A	83.77 ± 0.20	60.72 ± 0.50
LAME (RBF)	39.74 ± 1.17	22.85 ± 1.94
LAME (kNN)	82.60 ± 0.20	61.30 ± 0.46
AdaNPC	80.92 ± 1.38	61.68 ± 0.38
VarMatch (ours)	$\mathbf{86.32 \pm 0.20}$	$\mathbf{65.55 \pm 0.30}$
Updated parameters		
TENT	84.45 ± 0.53	62.74 ± 0.40
SHOT	$\underline{85.72 \pm 0.10}$	$\underline{65.46 \pm 0.25}$

4.2 Ablations

We retrain after removing one training component at a time (Table 3); combination rows drop more than one component. Full matches Table 1. Pixel-level augmentation is the only indispensable training component (-4.03 overall, -7.00 on Sketch). Removing any other single factor changes the PACS mean by at most 0.42, within seed variation. At 10 labels per class, removing center-pull or SupCon nominally improves PACS (86.63 and 86.67). At 5 labels, removing center-pull alone has little effect on PACS (85.70 vs. 85.46) and reduces OfficeHome by 0.42 points; removing both center-pull and SupCon reduces PACS from 85.46 to 85.04, driven by Sketch relative to the no-center-pull model (82.98 to 80.67), and has little effect on OfficeHome (62.26 vs. 62.31). We therefore retain both auxiliary terms and keep the same configuration across budgets.

The variational view. Removing VSR costs 0.42 points on PACS, smaller than the seed variation on Sketch alone. Target accuracy mixes this training effect with domain shift, which VSR is not designed to correct, so we also report source-side diagnostics from the same Full vs. −VSR runs. On the source validation split, VSR improves accuracy by 1.25 points (91.48 vs. 90.23, $t = 6.9$); on the unlabeled source set, it improves pseudo-label accuracy by 1.85 points and the retained fraction by 1.44 points, and raises the Fisher ratio from 2.055 to 2.453, a 19.4% gain in between-class separation relative to within-class spread. Appendix C shows both quantities across training. The small effect on target accuracy therefore does not indicate that VSR is ineffective; rather, its benefits are more directly reflected in source-side pseudo-label quality and class separation.

5 Conclusion

We presented VarMatch, a semi-supervised domain generalization method that combines standard input-space augmentation with an uncertainty-conditioned latent consistency view, and reuses the learned representation for frozen, source-aware test-time adaptation. Across the evaluated SSDG benchmarks and label budgets, VarMatch improves over the strongest compared SSL and SSDG training baselines. On PACS and OfficeHome, where we compare test-time methods on the same checkpoints at 10 labels per class, its frozen source-aware test-time procedure outperforms the parameter-free baselines we evaluate and remains competitive with methods that update network parameters. Further analysis shows that the latent view improves pseudo-label accuracy and class separation on unlabeled source data, while the test-time procedure benefits from source-aware latent alignment and proximity-filtered prototype adaptation. These results show that the variational posterior serves two roles: providing uncertainty-conditioned consistency during training and supporting source-aware adaptation at test time. Future work could explore latent perturbations drawn from sources independent of the encoder, and class-conditional priors.

AI USE STATEMENT

In this work, we used generative AI tools to assist with implementing parts of the experimental codebase, rephrasing and refining the text, identifying related work, and formatting LaTeX and mathematical expressions. We did not use generative AI tools to develop the methodology, formulate or prove mathematical claims, design the experiments, or generate experimental data. All AI-assisted work was reviewed and verified by the authors: code was checked against the reported results, and citations were verified against the corresponding published works. The authors take full responsibility for the final content of this work, including all text, claims, and artifacts produced with the assistance of generative AI.

REPRODUCIBILITY STATEMENT

To ensure the reproducibility of VarMatch, we provide comprehensive details regarding the datasets, network architectures, and hyperparameter selections in Appendix B and detailed pseudocode in Appendix A. Additionally, the complete source code is included in the supplementary material and will be made publicly available upon publication.

REFERENCES

Alexander A. Alemi, Ian Fischer, Joshua V. Dillon, and Kevin Murphy. Deep variational information bottleneck. In *International Conference on Learning Representations*, 2017.

Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019.

David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin Raffel. MixMatch: A holistic approach to semi-supervised learning. In *Advances in Neural Information Processing Systems*, pp. 5050–5060, 2019.

David Berthelot, Nicholas Carlini, Ekin D. Cubuk, Alex Kurakin, Kihyuk Sohn, Han Zhang, and Colin Raffel. ReMixMatch: Semi-supervised learning with distribution matching and augmentation anchoring. In *International Conference on Learning Representations*, 2020.

Malik Boudiaf, Romain Mueller, Ismail Ben Ayed, and Luca Bertinetto. Parameter-free online test-time adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8344–8353, 2022.

Junbum Cha, Sanghyuk Chun, Kyungjae Lee, Han-Cheol Cho, Seunghyun Park, Yunsung Lee, and Sungrae Park. SWAD: Domain generalization by seeking flat minima. In *Advances in Neural Information Processing Systems*, pp. 22405–22418, 2021.

Hao Chen, Ran Tao, Yue Fan, Yidong Wang, Jindong Wang, Bernt Schiele, Xing Xie, Bhiksha Raj, and Marios Savvides. SoftMatch: Addressing the quantity-quality trade-off in semi-supervised learning. In *International Conference on Learning Representations*, 2023.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pp. 1597–1607, 2020.

Yingjun Du, Jun Xu, Huan Xiong, Qiang Qiu, Xiantong Zhen, Cees G. M. Snoek, and Ling Shao. Learning to learn with variational information bottleneck for domain generalization. In *European Conference on Computer Vision*, pp. 200–216, 2020.

Chamuditha Jayanga Galappaththige, Sanoojan Baliah, Malitha Gunawardhana, and Muhammad Haris Khan. Towards generalizing to unseen domains with few labels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 23691–23700, 2024.

Zeyi Huang, Haohan Wang, Eric P. Xing, and Dong Huang. Self-challenging improves cross-domain generalization. In *European Conference on Computer Vision*, pp. 124–140, 2020.

Yusuke Iwasawa and Yutaka Matsuo. Test-time classifier adjustment module for model-agnostic domain generalization. In *Advances in Neural Information Processing Systems*, pp. 2427–2440, 2021.

Saeed Karimi and Hamdi Dibeklioglu. Adapac: Prototypical anchored contrastive test time adaptation for domain generalization. *Neurocomputing*, 650:130839, 2025.

Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. In *Advances in Neural Information Processing Systems*, pp. 18661–18673, 2020.

Diederik P. Kingma and Max Welling. Auto-encoding variational Bayes. In *International Conference on Learning Representations*, 2014.

Chia-Wen Kuo, Chih-Yao Ma, Jia-Bin Huang, and Zsolt Kira. FeatMatch: Feature-based augmentation for semi-supervised learning. In *European Conference on Computer Vision*, pp. 479–495, 2020.

Dongkwan Lee, Kyomin Hwang, and Nojun Kwak. Unlocking the potential of unlabeled data in semi-supervised domain generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 30599–30608, 2025.

Bo Li, Yifei Shen, Yezhen Wang, Wenzhen Zhu, Colorado Reed, Dongsheng Li, Kurt Keutzer, and Han Zhao. Invariant information bottleneck for domain generalization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 7399–7407, 2022a.

Xiaotong Li, Yongxing Dai, Yixiao Ge, Jun Liu, Ying Shan, and Ling-Yu Duan. Uncertainty modeling for out-of-distribution generalization. In *International Conference on Learning Representations*, 2022b.

Jian Liang, Dapeng Hu, and Jiashi Feng. Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation. In *International Conference on Machine Learning*, 2020.

Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Yaofo Chen, Shijian Zheng, Peilin Zhao, and Mingkui Tan. Efficient test-time model adaptation without forgetting. In *International Conference on Machine Learning*, volume 162, pp. 16888–16905, 2022.

Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Zhiquan Wen, Yaofo Chen, Peilin Zhao, and Mingkui Tan. Towards stable test-time adaptation in dynamic wild world. In *International Conference on Learning Representations*, 2023.

Genki Osada, Budrul Ahsan, Revoti Prasad Bora, and Takashi Nishide. Regularization with latent space virtual adversarial training. In *European Conference on Computer Vision*, pp. 565–581. Springer International Publishing, 2020.

Lei Qi, Hongpeng Yang, Yinghuan Shi, and Xin Geng. MultiMatch: Multi-task learning for semi-supervised domain generalization. *ACM Transactions on Multimedia Computing, Communications and Applications*, 20(6):184:1–184:21, 2024.

Kihyuk Sohn, David Berthelot, Nicholas Carlini, Zizhao Zhang, Han Zhang, Colin A. Raffel, Ekin Dogus Cubuk, Alexey Kurakin, and Chun-Liang Li. FixMatch: Simplifying semi-supervised learning with consistency and confidence. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

Naftali Tishby, Fernando C Pereira, and William Bialek. The information bottleneck method. In *Proceedings of the 37th Annual Allerton Conference on Communication, Control, and Computing*, pp. 368–377, 1999.

Vladimir N. Vapnik. *Statistical Learning Theory*. Wiley, 1998.

Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. In *International Conference on Learning Representations*, 2021.

Yidong Wang, Hao Chen, Qiang Heng, Wenxin Hou, Yue Fan, Zhen Wu, Jindong Wang, Marios Savvides, Takahiro Shinozaki, Bhiksha Raj, Bernt Schiele, and Xing Xie. FreeMatch: Self-adaptive thresholding for semi-supervised learning. In *International Conference on Learning Representations*, 2023.

Yulin Wang, Xuran Pan, Shiji Song, Hong Zhang, Gao Huang, and Cheng Wu. Implicit semantic data augmentation for deep networks. In *Advances in Neural Information Processing Systems*, volume 32, pp. 12614–12623, 2019.

Qizhe Xie, Zihang Dai, Eduard Hovy, Minh-Thang Luong, and Quoc V. Le. Unsupervised data augmentation for consistency training. In *Advances in Neural Information Processing Systems*, pp. 6256–6268, 2020.

Bowen Zhang, Yidong Wang, Wenxin Hou, Hao Wu, Jindong Wang, Manabu Okumura, and Takahiro Shinozaki. FlexMatch: Boosting semi-supervised learning with curriculum pseudo labeling. In *Advances in Neural Information Processing Systems*, pp. 18408–18419, 2021.

Yi-Fan Zhang, Xue Wang, Kexin Jin, Kun Yuan, Zhang Zhang, Liang Wang, Rong Jin, and Tieniu Tan. AdaNPC: Exploring non-parametric classifier for test-time adaptation. In *International Conference on Machine Learning*, pp. 41647–41676, 2023.

Kaiyang Zhou, Yongxin Yang, Yu Qiao, and Tao Xiang. Domain generalization with MixStyle. In *International Conference on Learning Representations*, 2021.

Kaiyang Zhou, Chen Change Loy, and Ziwei Liu. Semi-supervised domain generalization with stochastic StyleMatch. *International Journal of Computer Vision*, 131(9):2377–2387, 2023.

Sumaiya Zoha, Jeong-Gun Lee, and Young-Woong Ko. CAT: Class-aware adaptive-thresholding for robust semi-supervised domain generalization. *PLOS ONE*, 20(9):e0329799, 2025.

Supplementary Material

A Method Implementation Details

This section provides further implementation details for the methodology introduced in Section 3.

Classifier. In the stochastic classifier of Zhou et al. (2023), each class weight is sampled from a Gaussian centered at $\boldsymbol{w}_c$ with a learned per-dimension variance. We additionally move each $\boldsymbol{w}_c$ toward the mean sampled latent code of the labeled samples of class c in the batch, $\boldsymbol{w}_c \leftarrow \nu\, \boldsymbol{w}_c + (1-\nu)\, \bar{z}_c$ with $\nu = 0.95$, without backpropagation.

A.1 Latent Transformation

Algorithm 1 summarizes the procedure of Section 3.2 and Table 5 lists its constants. We note the edge cases and design choices not covered there.

Unlabeled source samples enter $\boldsymbol{m}_c^{\mathrm{src}}$ only when at least 3 pass the confidence threshold for class c, so that no centroid is dominated by one or two pseudo-labels. Target centroids are computed from the original $\boldsymbol{\mu}_j^{\mathrm{tgt}}$ rather than the shifted $\boldsymbol{\mu}'_j$, since Eq. 8 is applied to unshifted representations at inference. A class with between 5 and R assigned samples uses all of them; a class with fewer than 5 is marked invalid and its centroid defaults to the mean of all target representations.

The global update averages valid class centroids with equal weight, so that the unknown target class proportions do not bias the translation, and is skipped when fewer than 5 classes are valid. Convergence checks begin after iteration t_0, and convergence is declared once the change in $\|\bar{\boldsymbol{\mu}}^{\mathrm{tgt}}\|$ between iterations falls below ε.

Table 5: LT constants and their rationale.

Constant	Value	Rationale
Source confidence τ_{src}	0.95	Source-side cutoff for unlabeled samples entering the class statistics
Bank size per class N	100	Bounds memory and proximity cost
Target support size R	$\max(30, \lfloor M/3\lvert\mathcal{Y}\rvert \rfloor)$	Floor for a stable mean ($\lvert\mathcal{Y}\rvert$ = number of classes)
Update rate ρ	0.5	Exponential moving average update of global mean
Maximum iterations T	100	Safeguard
Convergence check start t_0	15	Convergence checks begin at this iteration
Tolerance ε	10^{-4}	Convergence criterion
Scale clamp $[s_{\min}, s_{\max}]$	$[0.5, 2]$	Prevents extreme rescaling of a class

A.2 TT-PI

Algorithm 2 summarizes the procedure of Section 3.3 and Table 6 lists its constants. We give the proximity score and the rationale for each constant.

Proximity score. Let $\mathcal{B}_c$ denote the class-c entries of the memory bank $\mathcal{B}$. For a target sample j,

$$\mathrm{prox}(j) = \max_c \log \sum_{\boldsymbol{b} \in \mathcal{B}_c} \exp\left(\frac{\cos(\tilde{\boldsymbol{\mu}}_j, \boldsymbol{b})}{T_{\mathrm{prox}}}\right). \tag{11}$$

The low temperature makes the score dominated by the most similar source exemplars. The threshold γ is computed separately for each target domain from its unlabeled target proximities alone. Samples with $\mathrm{prox}(j) > \gamma$ are excluded as source-redundant: including them would pull the target average back toward the source.

Algorithm 1 Latent Transformation.

Require: labeled source set $\mathcal{X}$ with class subsets $\mathcal{X}_c$, unlabeled source set $\mathcal{U}$, target images $\{x_j\}_{j=1}^{M}$, frozen encoder giving $\boldsymbol{\mu}(x) = f_\mu(f(x))$, frozen classifier C, source confidence threshold τ_{src}, bank size N, support size R, update rate ρ, maximum iterations T

Step 1: Source statistics

$\boldsymbol{\mu}_i = \boldsymbol{\mu}(x_i), \quad \hat{y}_i = \arg\max_c C(\boldsymbol{\mu}_i)_c, \quad \kappa_i = \max_c \text{softmax}(C(\boldsymbol{\mu}_i))_c$ for all source x_i

$\mathcal{U}_c = \{i \in \mathcal{U} \mid \hat{y}_i = c,\ \kappa_i \geq \tau_{\text{src}}\}$ ▷ confident unlabeled samples of class c

$\boldsymbol{m}_c^{\text{src}} = \text{mean}_{i \in \mathcal{X}_c \cup \mathcal{U}_c} \boldsymbol{\mu}_i, \quad \bar{\boldsymbol{\mu}}^{\text{src}} = \text{mean}_{i \in \mathcal{X} \cup \mathcal{U}} \boldsymbol{\mu}_i$

$\mathcal{B} = \mathcal{X} \cup \bigcup_c \mathcal{U}_c^N$, with $\mathcal{U}_c^N$ the N samples of $\mathcal{U}_c$ with largest κ_i ▷ used by TT-PI, Sec. 3.3

Step 2: Encode the target set once

$\boldsymbol{\mu}_j^{\text{tgt}} = \boldsymbol{\mu}(x_j)$ for $j = 1, \ldots, M; \quad \bar{\boldsymbol{\mu}}^{\text{tgt}} \leftarrow \frac{1}{M} \sum_{j=1}^{M} \boldsymbol{\mu}_j^{\text{tgt}}$

Step 3: Estimate target class centroids

for $t = 0$ **to** $T - 1$ **do**

  $\boldsymbol{\mu}'_j = \boldsymbol{\mu}_j^{\text{tgt}} - \bar{\boldsymbol{\mu}}^{\text{tgt}} + \bar{\boldsymbol{\mu}}^{\text{src}}$ ▷ shift into the source frame

  $\hat{y}_j = \arg\max_c C(\boldsymbol{\mu}'_j)_c, \quad \kappa_j = \max_c \text{softmax}(C(\boldsymbol{\mu}'_j))_c$

  $\mathcal{R}_c =$ the R indices j with $\hat{y}_j = c$ and largest κ_j

  $$\boldsymbol{m}_c^{\text{tgt}} = \frac{\sum_{j \in \mathcal{R}_c} \kappa_j \, \boldsymbol{\mu}_j^{\text{tgt}}}{\sum_{j \in \mathcal{R}_c} \kappa_j}$$ ▷ original, unshifted $\boldsymbol{\mu}_j^{\text{tgt}}$

  $\bar{\boldsymbol{\mu}}^{\text{tgt}} \leftarrow (1 - \rho)\, \bar{\boldsymbol{\mu}}^{\text{tgt}} + \rho\, \text{mean}_c \, \boldsymbol{m}_c^{\text{tgt}}$

  if converged **then break**

  end if

end for

Step 4: Class-wise scale

$$s_c = \frac{\text{median}_{i \in \mathcal{X}_c \cup \mathcal{U}_c} \|\boldsymbol{\mu}_i - \boldsymbol{m}_c^{\text{src}}\|}{\text{median}_{j \in \mathcal{R}_c} \|\boldsymbol{\mu}_j^{\text{tgt}} - \boldsymbol{m}_c^{\text{tgt}}\|}$$

Step 5: Inference (per target image, $\boldsymbol{\mu}$ averaged over the image and, when used, its flip)

$\hat{y} = \arg\max_c C(\boldsymbol{\mu})_c; \quad \tilde{\boldsymbol{\mu}} = s_{\hat{y}}(\boldsymbol{\mu} - \boldsymbol{m}_{\hat{y}}^{\text{tgt}}) + \boldsymbol{m}_{\hat{y}}^{\text{src}}$ ▷ Eq. 8

$\hat{y} = \arg\max_c C(\tilde{\boldsymbol{\mu}})_c; \quad \tilde{\boldsymbol{\mu}} = s_{\hat{y}}(\boldsymbol{\mu} - \boldsymbol{m}_{\hat{y}}^{\text{tgt}}) + \boldsymbol{m}_{\hat{y}}^{\text{src}}$ ▷ recomputed from the original $\boldsymbol{\mu}$

return $\hat{y}$ and κ from $C(\tilde{\boldsymbol{\mu}})$ ▷ pseudo-labels for TT-PI

Prototype weight α. With $\alpha = 1$, a class prototype is fully replaced by its target estimate wherever target support exists. Retaining part of the source prototype would keep some of the geometry TT-PI is intended to correct, and fixing α at the endpoint removes an interpolation degree of freedom.

Proximity quantile q. Excluding the most source-redundant fifth of each target domain removes samples whose similarity to the source is unusually strong relative to the rest of their own domain; those samples are left out of the target average, since including them would pull that average back toward the source. The same q is used for every benchmark, label budget and target domain.

Support threshold τ_{tt}. Training uses the stricter fixed threshold $\tau = 0.95$ where the adaptive gate is disabled, since false pseudo-labels produce gradient updates that can accumulate through confirmation bias. TT-PI instead averages retained samples into centroids, so a single wrong label shifts the prototype only slightly, making per-class coverage more important than per-sample purity. Because confidence typically decreases under domain shift, applying the training threshold at test time could leave insufficient support for the classes most affected by the shift. The proximity filter compensates for the lower threshold by excluding samples that are redundant with the source support.

Table 6: TT-PI constants and their rationale.

Constant	Value	Rationale
Prototype weight α	1.0	Avoids retaining source geometry TT-PI corrects
Proximity quantile q	0.8	Drops the most source-redundant fifth of the target
Support threshold τ_{tt}	0.90	Looser than training; centroid averaging attenuates errors
Proximity temperature T_{prox}	0.01	Low, so the score follows the nearest source exemplars

Algorithm 2 Test-time Prototype Interpolation (TT-PI).

Require: LT outputs $\tilde{\boldsymbol{\mu}}_j$, $\hat{y}_j$, κ_j for target images $\{x_j\}_{j=1}^{M}$; memory bank $\mathcal{B}$ with class subsets $\mathcal{B}_c$; labeled source set $\mathcal{X}$ with class subsets $\mathcal{X}_c$; frozen backbone f; constants τ_{tt}, q, α, T_{prox}

Step 1: Backbone features (averaged over the image and, when used, its flip)

$\boldsymbol{h}_i = f(x_i)$ for $i \in \mathcal{X}$; $\quad \boldsymbol{h}_j = f(x_j)$ for $j = 1, \ldots, M$

$\hat{\boldsymbol{m}}_c^{\mathrm{src}} = \mathrm{mean}_{i \in \mathcal{X}_c} \boldsymbol{h}_i$ ▷ source prototypes

Step 2: Support set

$\mathrm{prox}(j) = \max_c \log \sum_{\boldsymbol{b} \in \mathcal{B}_c} \exp\big(\cos(\tilde{\boldsymbol{\mu}}_j, \boldsymbol{b})/T_{\mathrm{prox}}\big)$ ▷ Eq. 11

$\gamma =$ the q-quantile of $\{\mathrm{prox}(j)\}_{j=1}^{M}$

$\mathcal{S} = \{j \mid \kappa_j \geq \tau_{\mathrm{tt}},\ \mathrm{prox}(j) \leq \gamma\}$; $\quad \mathcal{S}_c = \{j \in \mathcal{S} \mid \hat{y}_j = c\}$

Step 3: Prototype interpolation

$\hat{\boldsymbol{m}}_c^{\mathrm{tgt}} = \mathrm{mean}_{j \in \mathcal{S}_c} \boldsymbol{h}_j$

$\hat{\boldsymbol{m}}_c = (1-\alpha)\, \hat{\boldsymbol{m}}_c^{\mathrm{src}} + \alpha\, \hat{\boldsymbol{m}}_c^{\mathrm{tgt}}$ ▷ Eq. 10

Step 4: Prediction

return $y_j^* = \arg\max_c \cos(\boldsymbol{h}_j, \hat{\boldsymbol{m}}_c)$ for $j = 1, \ldots, M$

B Reproducibility Details

This section provides the experimental configurations used in Sec. 4; symbols follow the definitions in Sec. 3. Table 7 summarizes the benchmark-specific training hyperparameters, with all remaining settings shared across benchmarks.

We retain from the protocols of Zhou et al. (2023) on PACS and OfficeHome and Lee et al. (2025) on Digits-DG and miniDomainNet the leave-one-domain-out protocol, labeling budgets, ImageNet-pretrained ResNet-18 at 224×224, sequential per-domain sampling, and cosine learning-rate schedule. Because $\mathcal{L}_{\mathrm{labeled}}$ and $\mathcal{L}_{\mathrm{strong}}$ backpropagate through reparameterized draws, their gradients inherit posterior sampling variance. The published encoder rate of 0.003 was unstable early in training and caused collapse on several PACS, OfficeHome, and miniDomainNet runs, so we use lower encoder rates on these benchmarks and a classifier rate of 0.03; Digits-DG retains the published rates. We extend training to 150 epochs on PACS and OfficeHome and 80 on Digits-DG because source-validation accuracy is still rising at the end of the published schedule; on miniDomainNet, each epoch contains substantially more unlabeled updates, so we use 40 epochs, which exceeds the published 20. We average the weights of the final checkpoints (Table 7), avoiding source-validation-based checkpoint selection and reducing sensitivity to the choice of the final training epoch.

PACS uses batch size 30 rather than 48. The adaptive confidence threshold is used on PACS and Digits-DG, whereas OfficeHome and miniDomainNet use FixMatch's fixed $\tau = 0.95$ because their 65 and 126 classes provide too few labeled examples per class for reliable class-specific thresholds. Horizontal flipping is disabled for Digits-DG because it is not label-preserving. Training begins with a labeled-only warmup (Table 7), allowing the classifier to establish initial class structure before introducing pseudo-label-based consistency losses. The contrastive losses and, where used, the adaptive gate start later, at the epochs given in the same table. The KL weight β is cosine-annealed from 10^{-6} to the shared final value of 2×10^{-6} over the benchmark-specific periods in Table 7, to stabilize the variational bottleneck early in training.

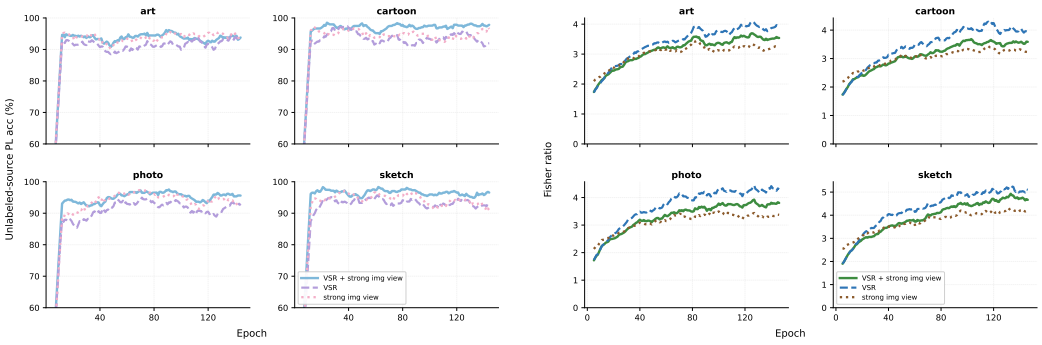
All remaining settings are shared: SGD with momentum 0.9, weight decay 5×10^{-3}, no learning-rate warmup, $T_{\text{cls}} = 0.05$, $T_{\text{proto}} = 0.07$, $\eta = 0.2$, EMA momentum of the adaptive threshold 0.9985, $T_{\text{con}} = 0.1$, $\lambda_{\text{con}} = 0.05$, $\lambda_{\text{pull}} = 0.02$, $\lambda_u = 1$, $s_0 = 4$, and $\omega = 2$.

All experiments were conducted on a system equipped with four NVIDIA GeForce RTX 3090 GPUs (24 GB each, driver 560.35.03), using PyTorch 2.5.1 with CUDA 12.1. Each leave-one-domain-out run was assigned to a single GPU, with runs across domains and random seeds executed in parallel.

Table 7: Per-benchmark training hyperparameters; all remaining settings are shared. For the adaptive confidence gate, the number is the epoch at which it is enabled.

	PACS	OfficeHome	Digits-DG	miniDomainNet
Classes	7	65	10	126
Labeled images (10 / 5 lab.)	210 / 105	1950 / 975	300 / 150	3780 / 1890
Image resolution	224	224	224	224
Encoder LR	5×10^{-4} [3]	3×10^{-4}	3×10^{-3}	3×10^{-4}
Classifier LR	0.03	0.03	0.01	0.03
Batch size (lab. = unlab.)	30	48	48	48
Confidence gate	adaptive (20)	$\tau = 0.95$	adaptive (10)	$\tau = 0.95$
Contrastive losses from epoch	20	20	10	5
β annealing epochs	60	60	10	16
Labeled-only warmup (epochs)	5	5	2	2
Horizontal flip (train / test)	✓ / ✓	✓ / ✓	– / –	✓ / ✓
Epochs	150	150	80	40
Checkpoint average	last 5	last 5	last 3	last 3

C Additional Analysis



(a) Pseudo-label accuracy

(b) Fisher ratio

Figure 3: Effect of the variational view on the unlabeled source set over training, one panel per PACS target domain at 10 labels per class, seed 2. Both quantities are computed on source data only and so are unaffected by domain shift. Curves are moving-average smoothed.

[3] On PACS, the Sketch target uses 9×10^{-4}; the other three targets use 5×10^{-4}. None of its three source domains contains line drawings, so a larger rate was used to allow further movement from the ImageNet-pretrained features.